

# Proceedings of the NASA Conference on Space Telerobotics

## Volume II

G. Rodriguez  
H. Seraji  
Editors

(NASA-CR-186897) PROCEEDINGS OF THE NASA  
CONFERENCE ON SPACE TELEROBOTICS, VOLUME 2  
(JPL) 393 p CSCL 054

NR0-29044  
--THRU--  
NR0-29079  
Unclass  
0295979

63/54

January 31, 1989



National Aeronautics and  
Space Administration

Jet Propulsion Laboratory  
California Institute of Technology  
Pasadena, California

1. The first step is to identify the problem or question that needs to be addressed. This involves understanding the context and the specific requirements of the task.

2. Next, it is important to gather relevant information and data. This can be done through research, consultation with experts, or by analyzing existing resources.

3. Once the information is gathered, the next step is to develop a plan or strategy. This involves breaking down the problem into smaller, manageable parts and determining the best approach to solve each part.

4. The fourth step is to implement the plan. This involves putting the strategy into action and monitoring progress along the way.

5. Finally, it is important to evaluate the results and make adjustments as needed. This involves reflecting on what worked well and what didn't, and using that information to improve future performance.

JPL Publication 89-7, Vol. II

# Proceedings of the NASA Conference on Space Telerobotics

Volume II

G. Rodriguez  
H. Seraji  
Editors

January 31, 1989



National Aeronautics and  
Space Administration

**Jet Propulsion Laboratory**  
California Institute of Technology  
Pasadena, California

This publication was prepared by the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.



ABSTRACT  
NASA Conference on Space Telerobotics

These proceedings contain papers presented at the NASA Conference on Space Telerobotics held in Pasadena, January 31-February 2, 1989. The Conference was sponsored by the NASA Office of Aeronautics and Space Technology, together with ARC, LRC, GSFC, JSC, MSFC, KSC and JPL. The theme of the Conference was man-machine collaboration in space. The Conference provided a forum for researchers and engineers to exchange ideas on the research and development required for application of telerobotics technology to the space systems planned for the 1990s and beyond. The Conference: (i) provided a view of current NASA telerobotic research and development; (ii) stimulated technical exchange on man-machine systems, manipulator control, machine sensing, machine intelligence, concurrent computation, and system architectures; and (iii) identified important unsolved problems of current interest which can be dealt with by future research. There were about 500 international participants including about 100 from abroad.

An international program committee was established for the conference. A.K. Bejczy and H. Seraji of JPL acted as co-chairs for this committee. Members of the committee were

J. Amat, University of Barcelona, Spain  
G.A. Bekey, University of Southern California  
P.R. Belanger, McGill University, Canada  
R.C. Bolles, Stanford Research Center  
J.G. Bollinger, University of Wisconsin  
W.J. Book, Georgia Institute of Technology  
J.M. Brady, Oxford University, UK  
F.E.C. Culick, California Institute of Technology  
R.J.P. deFigueiredo, Rice University  
W.R. Ferrell, University of Arizona  
E. Freund, University of Dortmund, FRG  
A.A. Goldenberg, University of Toronto, Canada  
R. Jain, University of Michigan  
T. Kanade, Carnegie-Mellon University  
I. Kato, Waseda University, Japan  
A.J. Koivo, Purdue University  
P.D. Lawrence, University of British Columbia  
J.Y.S. Luh, Clemson University  
H.E. Rauch, Lockheed Palo Alto Research Lab  
A. Rovetta, Polytechnic University of Milan  
G.N. Saridis, Rensselaer Polytechnic Institute  
T.B. Sheridan, Massachusetts Institute of Technology  
L. Stark, University of California, Berkeley  
D. Tesar, University of Texas at Austin  
H. Van Brussel, Catholic University of Leuven  
R.A. Volz, Texas Tech University

The Conference was organized by the Telerobotics Working Group of the NASA Office of Aeronautics and Space Technology. M. Montemerlo of NASA Headquarters and S.Z. Szirmay co-chair this working group. Representatives to this group from NASA centers and other research organizations are

- D. Akin, Massachusetts Institute of Technology
- J. Bull, Ames Research Center
- R. Davis, Kennedy Space Center
- S. Fisher, Ames Research Center
- J. Haussler, Marshall Space Flight Center
- A. Meintel, Langley Research Center
- J. Pennington, Langley Research Center
- D. Provost, Goddard Space Flight Center
- C. Price, Johnson Space Center
- L. Purves, Goddard Space Flight Center
- C. Ruoff, Jet Propulsion Laboratory
- E.C. Smith, Marshall Space Flight Center
- M. Zweben, Ames Research Center

## ACKNOWLEDGMENTS

Acknowledgments are due to R. Doshi, D. Diner, J. Barhen and A. Fijany of JPL and Professors L. Stark of UCB and H. Stephanou of George Mason University for their help in organizing invited technical sessions and discussion panels. Appreciation is due to P. McLane for setting up registration and conference facilities, and to L. Anderson for technical editing of the proceedings. Acknowledgments are also due Donna L. Milton of JPL for handling the local arrangements and coordination and administrative aspects of the Conference.



# CONTENTS

## Volume I

OPENING SESSION . . . . .	1
Remarks Made at the Beginning of the NASA Conference on Space Telerobotics . . . . .	3
G. Varsi . . . . .	
Conference Welcome . . . . .	5
T.E. Everhart . . . . .	
Evolving Space Teleoperation to Space Telerobotics: Research and Systems Considerations . . . . .	7
M. Montemerlo . . . . .	
Space Telerobotics Conference Objectives . . . . .	15
A.K. Bejczy . . . . .	
REDUNDANT MANIPULATORS 1 . . . . .	17
A 17 Degree of Freedom Anthropomorphic Manipulator H.I. Vold, J.P. Karlen, J.M. Thompson, J.D. Farrell, and P.H. Eismann . . . . .	19
A New Approach to Global Control of Redundant Manipulators H. Seraji . . . . .	29
Kinematic Functions for the 7 DOF Robotics Research Arm K. Kreutz, M. Long, and H. Seraji . . . . .	39
Cartesian Control of Redundant Robots R. Colbaugh and K. Glass . . . . .	49
Kinematics, Controls, and Path Planning Results for a Redundant Manipulator . . . . .	59
B. Gretz and S. Tilley . . . . .	
A Complete Analytical Solution for the Inverse Instantaneous Kinematics of a Spherical-Revolute-Spherical (7R) Redundant Manipulator . . . . .	69
R.P. Podhorodeski, R.G. Fenton, and A.A. Goldenberg . . . . .	
MAN-MACHINE SYSTEMS . . . . .	79
Adjustable Impedance, Force Feedback and Command Language Aids for Telerobotics . . . . .	
T.B. Sheridan, G.J. Raju, F.T. Buzan, W. Yared, and J. Park . . . . .	81

Variable Force and Visual Feedback Effects on Teleoperator Man/Machine Performance M.J. Massimino and T.B. Sheridan . . . . .	89
Teleoperator Comfort and Psychometric Stability: Criteria for Limiting Master-Controller Forces of Operation and Feedback During Telemanipulation S.F. Wiker, E. Hershkowitz, and J. Zik . . . . .	99
Measurement of Hand Dynamics in a Microsurgery Environment: Preliminary Data in the Design of a Bimanual Telemicro-Operation Test Bed S. Charles and R. Williams . . . . .	109
Human Factors Model Concerning the Man-Machine Interface of Mining Crewstations J.P. Rider and R.L. Unger . . . . .	119
Development of a Flexible Test-Bed for Robotics, Telemanipulation and Servicing Research B.F. Davies . . . . .	129
TELEROBOT ARCHITECTURES . . . . .	139
Control of Intelligent Robots in Space E. Freund and C. Bühler . . . . .	141
Modularity in Robotic Systems D. Tesar and M.S. Butler . . . . .	151
A System Architecture for a Planetary Rover D.B. Smith and J.R. Matijevic . . . . .	163
The NASA/OAST Telerobot Testbed Architecture J.R. Matijevic, W.F. Zimmerman, and S. Dolinsky . . . . .	185
Formulation of Design Guidelines for Automated Robotic Assembly in Outerspace S.N. Dwivedi, G. Jones, S. Banerjee, and S. Srivastava . . . . .	197
Automation and Robotics Technology for Intelligent Mining Systems J.H. Welsh . . . . .	207
ROBOT SENSING AND PLANNING . . . . .	217
A Fast Lightstripe Ranging System with Smart VLSI Sensor A. Gruss, L.R. Carley, and T. Kanade . . . . .	219
Methods and Strategies of Object Localization L. Shao and R.A. Volz . . . . .	229

A Laser Tracking Dynamic Robot Metrology Instrument G.A. Parker and J.R.R. Mayer . . . . .	241
Robot Acting on Moving Bodies (RAMBO): Interaction with Tumbling Objects L.S. Davis, D. DeMenthon, T. Bestul, S. Ziavras, H.V. Srinivasan, M. Siddalingaiah, and D. Harwood . . . . .	251
Real-Time Edge Tracking Using a Tactile Sensor A.D. Berger, R. Volpe, and P.K. Khosla . . . . .	261
Planning 3-D Collision-Free Paths Using Spheres S. Bonner and R.B. Kelley . . . . .	273
NAVIGATION . . . . .	283
Map Learning with Indistinguishable Locations K. Basye and T. Dean . . . . .	285
Three-dimensional Motor Schema Based Navigation R.C. Arkin . . . . .	291
Periodic Gaits for the CMU Ambler S. Mahalingam and S.N. Dwivedi . . . . .	301
Exploiting Map Plans as Resources for Action D. Payton . . . . .	311
Learned Navigation in Unknown Terrains: A Retraction Method N.S.V. Rao, N. Stoltzfus, and S.S. Iyengar . . . . .	321
NEURAL NETWORKS . . . . .	331
"Computational" Neural Learning Formalisms for Manipulator Inverse Kinematics S. Gulati, J. Barhen, and S.S. Iyengar . . . . .	333
Multi-Layer Neural Networks for Robot Control F. Pourboghraat . . . . .	343
A Hybrid Architecture for the Implementation of the Athena Neural Net Model C. Koutsougeras and C. Papachristou . . . . .	353
A Design Philosophy for Multi-Layer Neural Networks With Applications to Robot Control N. Vadiiee and M. Jamshidi . . . . .	363
A Neural Network for Controlling the Configuration of Frame Structure With Elastic Members K. Tsutsumi . . . . .	373

FUNDAMENTAL AI RESEARCH . . . . .	383
Coordinating the Activities of a Planner and an Execution Agent A. Tate . . . . .	385
Plan Recognition for Space Telerobotics B.A. Goodman and D.J. Litman . . . . .	395
Causal Simulation and Sensor Planning in Predictive Monitoring R.J. Doyle . . . . .	405
State-Based Scheduling: An Architecture for Telescope Observation Scheduling N. Muscettola and S.F. Smith . . . . .	415
Focus of Attention in an Activity-Based Scheduler N. Sadeh and M.S. Fox . . . . .	425
REASONING UNDER UNCERTAINTY . . . . .	435
A Boltzmann Machine for the Organization of Intelligent Machines M.C. Moed and G.N. Saridis . . . . .	437
Grasp Planning Under Uncertainty A.M. Erkmen and H.E. Stephanou . . . . .	447
Approximation Algorithms for Planning and Control M. Boddy and T. Dean . . . . .	457
Multiresolutional Models of Uncertainty Generation and Reduction A. Meystel . . . . .	463

## VOLUME II

REDUNDANT MANIPULATORS 2 . . . . .	1
Characterization and Control of Self-motions in Redundant Manipulators J. Burdick and H. Seraji . . . . .	3
Multiple Cooperating Manipulators: The Case of Kinematically Redundant Arms I.D. Walker, R.A. Freeman, and S.I. Marcus . . . . .	15
Reflexive Obstacle Avoidance for Kinematically-Redundant Manipulators J.P. Karlen, J.M. Thompson, Jr., J.D. Farrell, and H.I. Vold . . . .	25
Preliminary Study of a Serial-Parallel Redundant Manipulator V. Hayward and R. Kurtz . . . . .	39



TELEOPERATION 1 . . . . .	49
The JPL Telerobot Operator Control Station: Part I - Hardware E.P. Kan, J.T. Tower, G.W. Hunka, and G.J. VanSant . . . . .	51
The JPL Telerobot Operator Control Station: Part II - Software E.P. Kan, B.P. Landell, S. Oxenberg, and C. Morimoto . . . . .	63
Design of a Monitor and Simulation Terminal (Master) for Space Station Telerobotics and Telescience L. Lopez, C. Konkel, P. Harmon, and S. King . . . . .	75
Performance Evaluation of a 6 Axis High Fidelity Generalized Force Reflecting Teleoperator B. Hannaford and L. Wood . . . . .	87
Implementation and Design of a Teleoperation System Based on a VMEbus/68020 Pipelined Architecture T.S. Lee . . . . .	97
Human/Machine Interaction via the Transfer of Power and Information Signals H. Kazerooni, W.K. Foslien, B.J. Anderson, and T.M. Hessburg . . . .	109
TELEROBOTS 1 . . . . .	121
Trajectory Generation for Space Telerobots R. Lumia and A.J. Wavering . . . . .	123
On the Simulation of Space Based Manipulators with Contact M.W. Walker and J. Dionise . . . . .	133
Preliminary Results on Noncollocated Torque Control of Space Robot Actuators S.W. Tilley, C.M. Francis, K. Emerick, and M.G. Hollars . . . . .	143
Portable Dextrous Force Feedback Master for Robot Telemanipulation (P.D.M.F.F.) G.C. Burdea and T.H. Speeter . . . . .	153
Experiences with the JPL Telerobot Testbed - Issues and Insights H.W. Stone, B. Balaram, and J. Beahan . . . . .	163
The KALI Multi-Arm Robot Programming and Control Environment P. Backes, S. Hayati, V. Hayward, and K. Tso . . . . .	173
TELEROBOT PERCEPTION . . . . .	183
How Do Robots Take Two Parts Apart? R.K. Bajcsy and C.J. Tsikos . . . . .	185
Techniques and Potential Capabilities of Multi-Resolutional Information (Knowledge) Processing A. Meystel . . . . .	197

Perceptual Telerobotics	
P.A. Ligomenides . . . . .	211
Building an Environment Model Using Depth Information	
Y. Roth-Tabak and R. Jain . . . . .	221
ROVERS . . . . .	231
HERMIES-III: A Step Toward Autonomous Mobility, Manipulation and Perception	
C.R. Weisbin, B.L. Burks, J.R. Einstein, R.R. Feezell, W.W. Manges, and D.H. Thompson . . . . .	233
First Results in Terrain Mapping for a Roving Planetary Explorer	
E. Krotkov, C. Caillas, M. Hebert, I.S. Kweon, and T. Kanade . . . . .	247
Planetary Rover Technology Development Requirements	
R.J. Bedard, Jr., B.K. Muirhead, M.D. Montemerlo, and M.S. Hirschbein . . . . .	257
Rice-Obot I: An Intelligent Autonomous Mobile Robot	
R. deFigueiredo, L. Cisson, and D. Berberian . . . . .	265
Satellite-Map Position Estimation for the Mars Rover	
A. Hayashi and T. Dean . . . . .	275
Robotic Sampling System for an Unmanned Mars Mission	
W. Chun . . . . .	283
PARALLEL PROCESSING . . . . .	293
Efficient Mapping Algorithms for Scheduling Robot Inverse Dynamics Computation on a Multiprocessor System	
C.S.G. Lee and C.L. Chen . . . . .	295
Parallel Algorithms for Computation of the Manipulator Inertia Matrix	
M. Amin-Javaheri and D.E. Orin . . . . .	307
SPATIAL REPRESENTATIONS AND REASONING . . . . .	317
Planning Robot Actions Under Position and Shape Uncertainty	
C. Laugier . . . . .	319
Organising Geometric Computations for Space Telerobotics	
S. Cameron . . . . .	331
A Tessellated Probabilistic Representation for Spatial Robot Perception and Navigation	
A. Elfes . . . . .	341

NASA AMES RESEARCH CENTER . . . . .	351
A Survey of Planning and Scheduling Research at the NASA Ames Research Center	
M. Zweben . . . . .	353
Integrating Planning and Reactive Control	
S.J. Rosenschein and L.P. Kaelbling . . . . .	359
Learning in Stochastic Neural Networks for Constraint Satisfaction Problems	
M.D. Johnston and H.-M. Adorf . . . . .	367
Integrating Planning, Execution, and Learning	
D.R. Kuokka . . . . .	377

### VOLUME III

PLENARY SESSION . . . . .	1
The Flight Telerobotic Servicer: NASA's First Operational Space Robot	
C.F. Fuechsel . . . . .	3
FLEXIBLE ARMS . . . . .	9
Modeling, Design, and Control of Flexible Manipulator Arms: Status and Trends	
W.J. Book . . . . .	11
Dynamical Modeling of Serial Manipulators with Flexible Links and Joints Using the Method of Kinematic Influence	
P.L. Graves . . . . .	25
Capture of Free-Flying Payloads With Flexible Space Manipulators	
T. Komatsu, M. Uenohara, S. Iikura, H. Miura, and I. Shimoyama . .	35
Technology and Task Parameters Relating to the Effectiveness of the Bracing Strategy	
W.J. Book and J.J. Wang . . . . .	45
Manipulators with Flexible Links: A Simple Model and Experiments	
I. Shimoyama and I.J. Oppenheim . . . . .	59
Experiments in Identification and Control of Flexible-Link Manipulators	
S. Yurkovich, A.P. Tzes, and F.E. Pacheco . . . . .	69
ROBOTIC END-EFFECTORS AND HAND CONTROLLERS . . . . .	79
Autonomous Dexterous End-Effectors for Space Robotics	
G.A. Bekey, T. Iberall, and H. Liu . . . . .	81

Design and Control of a Multi-Fingered Robot Hand Provided With Tactile Feedback H. Van Brussel, B. Santoso, and D. Reynaerts . . . . .	89
Traction-Drive Force Transmission for Telerobotic Joints D.P. Kuban and D.M. Williams . . . . .	103
Force/Torque and Tactile Sensors for Sensor-Based Manipulator Control H. Van Brussel, H. Beliën, and C.-Y. Bao . . . . .	117
Redundant Sensorized Arm + Hand System for Space Telerobotized Manipulation A. Rovetta and P. Cavestro . . . . .	129
Impedance Hand Controllers for Increasing Efficiency in Teleoperations C. Carignan and J. Tarrant . . . . .	135
TELE-AUTONOMOUS SYSTEMS . . . . .	145
Tele-Autonomous Systems: New Methods for Projecting and Coordinating Intelligent Action at a Distance L. Conway, R. Volz, and M.W. Walker . . . . .	147
An Advanced Telerobotic System for Shuttle Payload Changeout Room Processing Applications M. Sklar, D. Wegerif, and L. Davis . . . . .	159
Robotic Tele-Existence S. Tachi, H. Arai, and T. Maeda . . . . .	171
Redundancy of Space Manipulator on Free-Flying Vehicle and Its Nonholonomic Path Planning Y. Nakamura and R. Mukherjee . . . . .	181
Guidance Algorithms for a Free-Flying Space Robot A.F. Brindle, H.E.M. Viggh, and J.H. Albert . . . . .	191
Telepresence System Development for Application to the Control of Remote Robotic Systems C.D. Crane III, J. Duffy, R. Vora, and S.-C. Chiang . . . . .	201
ROBOTIC VISION . . . . .	211
3D Model Control of Image Processing A.H. Nguyen and L. Stark . . . . .	213
Weighted Feature Selection Criteria for Visual Servoing of a Telerobot J.T. Feddema, C.S.G. Lee, and O.R. Mitchell . . . . .	223

Trinocular Stereovision using Figural Continuity, Dealing with Curved Objects	235
R. Vaillant and O.D. Faugeras . . . . .	
A Fast 3-D Object Recognition Algorithm for the Vision System of a Special-Purpose Dexterous Manipulator	245
S.H.Y. Hung . . . . .	
Use of 3D Vision for Fine Robot Motion	255
A. Lokshin and T. Litwin . . . . .	
TELEROBOTS 2 . . . . .	263
Telerobotic Workstation Design Aid	265
K. Corker, E. Hudlicka, D. Young, and N. Cramer . . . . .	
Space Robotic System for Proximity Operations	277
P.G. Magnani and M. Colomba . . . . .	
Modeling and Sensory Feedback Control for Space Manipulators	287
Y. Masutani, F. Miyazaki, and S. Arimoto . . . . .	
Control Strategies for a Telerobot	297
J. O'Hara and B. Stasi . . . . .	
Autonomous Sensor-Based Dual-Arm Satellite Grappling	307
B. Wilcox, K. Tso, T. Litwin, S. Hayati, and B. Bon . . . . .	
Thread: A Programming Environment for Interactive Planning-level Robotics Applications	317
J.J. Beahan, Jr. . . . .	
MULTI-ARM CONTROL . . . . .	329
Stability Analysis of Multiple-Robot Control Systems	331
J.T. Wen and K. Kreutz . . . . .	
Experiments in Cooperative Manipulation: A System Perspective	341
S.A. Schneider and R.H. Cannon, Jr. . . . .	
On the Manipulability of Dual Cooperative Robots	351
P. Chiacchio, S. Chiaverini, L. Sciavicco, and B. Siciliano . . . . .	
Controlling Multiple Manipulators Using RIPS	361
Y. Wang, S. Jordan, A. Mangaser, and S. Butner . . . . .	
Time Optimal Movement of Cooperating Robots	371
J.M. McCarthy and J.E. Bobrow . . . . .	

COUPLING OF SYMBOLIC AND NUMERIC SYSTEMS . . . . .	381
Reflections on the Relationship Between Artificial Intelligence and Operations Research M.S. Fox . . . . .	383
What Kind of Computation Is Intelligence? A Framework for Integrating Different Kinds of Expertise B. Chandrasekaran . . . . .	395
A Design Strategy for Autonomous Systems P. Forster . . . . .	403
Learning in Tele-autonomous Systems using Soar J.E. Laird, E.S. Yager, C.M. Tuck, and M. Hucka . . . . .	415
Design of a Structural and Functional Hierarchy for Planning and Control of Telerobotic Systems L. Acar and Ü. Özgüner . . . . .	425
NASA GODDARD SPACE FLIGHT CENTER . . . . .	435
The Flight Telerobotic Servicer Project: A Technical Overview H.G. McCain . . . . .	437
The Flight Telerobotic Servicer Tinman Concept: System Design Drivers and Task Analysis J.F. Andary, D.R. Hewitt, and S.W. Hinkal . . . . .	447
The Flight Telerobotic Servicer: From Functional Architecture to Computer Architecture R. Lumia and J. Fiala . . . . .	473
Research and Development Activities at the Goddard Space Flight Center for the Flight Telerobotic Servicer Project S. Ollendorf . . . . .	483
The Goddard Space Flight Center (GSFC) Robotics Technology Testbed R. Schnurr, M. O'Brien, and S. Cofer . . . . .	491
Test and Validation for Robot Arm Control Dynamics Simulation K.H. Yae, S.-S. Kim, E.J. Haug, W. Seering, K. Sundaram, B. Thompson, J. Turner, H. Chun, H.P. Frisch, and R. Schnurr . . . .	501
PANEL ON GRAPHIC OVERLAYS IN TELEOPERATION . . . . .	509
Graphic Overlays in High-Precision Teleoperation: Current and Future Work at JPL D.B. Diner and S.C. Venema . . . . .	511
Head-Mounted Spatial Instruments II: Synthetic Reality or Impossible Dream S.R. Ellis and A. Grunwald . . . . .	521

Use of Graphics in Decision Aids for Telerobotic Control	
T.B. Sheridan, J.B. Roseborough, H. Das, K.-P. Chin,	
and S. Inoue . . . . .	533

# VOLUME IV

MANIPULATOR CONTROL 1 . . . . .	1
An Improved Adaptive Control for Repetitive Motion of Robots	
F. Pourboghrat . . . . .	3
Direct Adaptive Control of a PUMA 560 Industrial Robot	
H. Seraji, T. Lee, and M. Delpech . . . . .	11
Model Based Manipulator Control	
L.J. Petrosky and I.J. Oppenheim . . . . .	23
Discrete-Time Adaptive Control of Robot Manipulators	
M. Tarokh . . . . .	33
A Discrete Decentralized Variable Structure Robotic Controller	
Z.S. Tumei . . . . .	43
TELEMANIPULATION . . . . .	53
Construction and Demonstration of a 9-String 6 DOF	
Force Reflecting Joystick for Telerobotics	
R. Lindemann and D. Tesar . . . . .	55
Response to Reflected-Force Feedback to Fingers in Teleoperations	
P.H. Sutter, J.C. Iatridis, and N.V. Thakor . . . . .	65
The Jau-JPL Anthropomorphic Telerobot	
B.M. Jau . . . . .	75
A Procedure Concept for Local Reflex Control of Grasping	
P. Fiorini and J. Chang . . . . .	81
Performance Limitations of Bilateral Force Reflection Imposed	
by Operator Dynamic Characteristics	
J.D. Chapel . . . . .	91
Sensor-based Fine Telemanipulation for Space Robotics	
M. Andrenucci, M. Bergamasco, and P. Dario . . . . .	101
FLIGHT EXPERIMENTS: SYSTEMS AND SIMULATORS . . . . .	109
ROTEX-TRIIFEX: Proposal for a Joint FRG-USA Telerobotic Flight	
Experiment	
G. Hirzinger and A.K. Bejczy . . . . .	111

Test and Training Simulator for Ground-Based Teleoperated In-Orbit Servicing B.E. Schäfer . . . . .	125
Concept Synthesis of an Equipment Manipulation and Transportation System (EMATS) W. De Peuter and E. Waffenschmidt . . . . .	135
Force-Reflective Teleoperated System With Shared and Compliant Control Capabilities Z. Szakaly, W.S. Kim, and A.K. Bejczy . . . . .	145
Information management in an Integrated Space Telerobot S. Di Pippo, G. Pasquariello, and G.S. Labini . . . . .	157
Redundancy in Sensors, Control and Planning of a Robotic System for Space Telerobotics A. Rovetta, S. Vodret, and M. Bianchini . . . . .	167
SENSOR-BASED PLANNING . . . . .	171
How to Push a Block Along a Wall M.T. Mason . . . . .	173
Global Models: Robot Sensing, Control, and Sensory-Motor Skills P.S. Schenker . . . . .	183
3-D Vision System Integrated Dexterous Hand R.C. Luo and Y.-S. Han . . . . .	187
A Layered Abduction Model of Perception: Integrating Bottom-up and Top-down Processing in a Multi-Sense Agent J.R. Josephson . . . . .	197
RCTS: A Flexible Environment for Sensor Integration and Control of Robot Systems - The Distributed Processing Approach R. Allard, B. Mack, and M.M. Bayoumi . . . . .	207
Vehicle Path-Planning in Three Dimensions Using Optics Analogs for Optimizing Visibility and Energy Cost N.C. Rowe and D.H. Lewis . . . . .	217
SPECIAL TOPICS . . . . .	227
Vacuum Mechatronics S. Hackwood, S.E. Belinski, and G. Beni . . . . .	229
Uniform Task Level Definitions for Robotic System Performance Comparisons C. Price and D. Tesar . . . . .	241



Linear Analysis of a Force Reflective Teleoperator K.B. Biggers, S.C. Jacobsen, and C.C. Davis . . . . .	245
Real-Time Cartesian Force Feedback Control of a Teleoperated Robot P. Campbell . . . . .	255
Optimal Payload Rate Limit Algorithm for Zero-G Manipulators M.L. Ross and D.A. McDermott . . . . .	263
Assembly of Objects With Not Fully Predefined Shapes M.A. Arlotti and V. Di Martino . . . . .	273
ROBOT KINEMATICS, DYNAMICS AND CONTROL . . . . .	283
Recursive Multibody Dynamics and Discrete-Time Optimal Control G.M.T. D'Eleuterio and C.J. Damaren . . . . .	285
The Effects of Gear Reduction on Robot Dynamics J. Chen . . . . .	297
Recursive Newton-Euler Formulation of Manipulator Dynamics M.G. Nasser . . . . .	309
Kinematic Sensitivity of Robot Manipulators M.I. Vuskovic . . . . .	319
Efficient Conjugate Gradient Algorithms for Computation of the Manipulator Forward Dynamics A. Fijany and R.E. Scheid . . . . .	329
On the Stability of Robotic Systems with Random Communication Rates H. Kobayashi, X. Yun, and R.P. Paul . . . . .	341
ROBOT TASK PLANNING AND ASSEMBLY . . . . .	351
Precedence Relationship Representations of Mechanical Assembly Sequences L.S. Homem de Mello and A.C. Sanderson . . . . .	353
Using Multiple Sensors for Printed Circuit Board Insertion D. Sood, M.C. Repko, and R.B. Kelley . . . . .	363
Determining Robot Actions For Tasks Requiring Sensor Interaction J. Budenske and M. Gini . . . . .	373
NASA LANGLEY RESEARCH CENTER . . . . .	383
The Laboratory Telerobotic Manipulator Program J.N. Herndon, S.M. Babcock, P.L. Butler, H.M. Costello, R.L. Glassell, R.L. Kress, D.P. Kuban, J.C. Rowe, and D.M. Williams . . . . .	385

Robotic Control of the Seven-Degree-of-Freedom NASA Laboratory Telerobotic Manipulator R.V. Dubey, J.A. Euler, R.B. Magness, S.M. Babcock, and J.N. Herndon . . . . .	395
The Control of Space Manipulators Subject to Spacecraft Attitude Control Saturation Limits S. Dubowsky, E.E. Vance, and M.A. Torres . . . . .	409
System Architectures for Telerobotic Research F.W. Harrison . . . . .	419
Comparison of Joint Space Versus Task Force Load Distribution Optimization for a Multiarm Manipulator System D.I. Soloway and T.E. Alberts . . . . .	431

## VOLUME V

PLENARY SESSION . . . . .	1
Telerobotic Activities at Johnson Space Center C.R. Price . . . . .	3
ROBOT ARM MODELING AND CONTROL . . . . .	9
Application of Recursive Manipulator Dynamics to Hybrid Software/Hardware Simulation C.J. Hill, K.A. Hopping, and C.R. Price . . . . .	11
Kinematics & Control Algorithm Development and Simulation for a Redundant Two-Arm Robotic Manipulator System M.P. Hennessey, P.C. Huang, and C.T. Bunnell . . . . .	21
Inverse Dynamics of a 3 Degree of Freedom Spatial Flexible Manipulator E. Bayo and M. Serna . . . . .	31
A Control Approach for Robots With Flexible Links and Rigid End-Effectors E. Barbieri and Ü. Özgüner . . . . .	41
SPECIAL TOPICS IN TELEOPERATION . . . . .	51
Preshaping Command Inputs to Reduce Telerobotic System Oscillations N.C. Singer and W.P. Seering . . . . .	53
Performance Constraints and Compensation For Teleoperation With Delay J.S. McLaughlin and B.D. Staunton . . . . .	63

Flight Telerobotic Servicer Control From the Orbiter T.M. Ward and D.L. Harlan . . . . .	73
Teleoperation Experiments with a Utah/MIT Hand and a VPL DataGlove D. Clark, J. Demmel, J. Hong, G. Lafferriere, L. Salkind, and X. Tan . . . . .	81
Instruction Dialogues: Teaching New Skills to a Robot C. Crangle and P. Suppes . . . . .	91
Interaset: A Natural Language Interface for Teleoperated Robotic Assembly of the EASE Space Structure D.K. Boorsma . . . . .	103
TELEROBOTIC SPACE OPERATIONS . . . . .	109
Establishing Viable Task Domains for Telerobot Demonstrations W. Zimmerman . . . . .	111
The Telerobot Workstation Testbed for the Shuttle Aft Flight Deck: A Project Plan for Integrating Human Factors into System Design T. Sauerwein . . . . .	121
Multi-Level Manual and Autonomous Control Superposition for Intelligent Telerobot S. Hirai and T. Sato . . . . .	131
An Alternative Control Structure for Telerobotics P.T. Boissiere and R.W. Harrigan . . . . .	141
Integration of a Sensor Based Multiple Robot Environment for Space Applications: The Johnson Space Center Teleoperator Branch Robotics Laboratory J. Hwang, P. Campbell, M. Ross, C.R. Price, and D. Barron . . . . .	151
MANIPULATOR CONTROL 2 . . . . .	161
Requirements for Implementing Real-Time Control Functional Modules on a Hierarchical Parallel Pipelined System T.E. Wheatley, J.L. Michaloski, and R. Lumia . . . . .	163
The JPL Telerobot Manipulator Control and Mechanization Subsystem (MCM) S. Hayati, T. Lee, K. Tso, P. Backes, E. Kan, and J. Lloyd . . . . .	173
On Discrete Control of Nonlinear Systems With Applications to Robotics M. Eslami . . . . .	183
A Spatial Operator Algebra for Manipulator Modeling and Control G. Rodriguez, K. Kreutz, and A. Jain . . . . .	193

FLIGHT EXPERIMENT CONCEPTS . . . . .	205
Flight Experiments in Telerobotics - Orbiter Middeck Concept L.M. Jenkins . . . . .	207
Experimental Study on Two-Dimensional Free-Flying Robot Satellite Model Y. Umetani and K. Yoshida . . . . .	215
The Astronaut and the Banana Peel: an EVA Retriever Scenario D.G. Shapiro . . . . .	225
Computed Torque Control of a Free-Flying Cooperating-Arm Robot R. Koningstein, M. Ullman, and R.H. Cannon, Jr. . . . .	235
Next Generation Space Robot T. Iwata, M. Oda, and R. Imai . . . . .	245
MANIPULATOR COORDINATION . . . . .	253
Coordination in a Hierarchical Multi-Actuator Controller A. Meystel . . . . .	255
Distributed Communications and Control Network for Robotic Mining W.H. Schiffbauer . . . . .	263
Computer Simulation and Design of a Three Degree-of-Freedom Shoulder Module D. Marco, L. Torfason, and D. Tesar . . . . .	273
A Collision Avoidance System for a Spaceplane Manipulator Arm A. Sciomachen and P.G. Magnani . . . . .	283
ISSUES IN AI SYSTEMS . . . . .	293
Generic Task Problem-Solvers in Soar T.R. Johnson, J.W. Smith, Jr., and B. Chandrasekaran . . . . .	295
Temporal Logics Meet Telerobotics E. Rutten and L. Marcé . . . . .	301
An Efficient Temporal Logic for Robotic Task Planning J.M. Becker . . . . .	311
The Indexed Time Table Approach for Planning and Acting M. Ghallab and A.M. Alaoui . . . . .	321
Reactive Behavior, Learning, and Anticipation S.D. Whitehead and D.H. Ballard . . . . .	333

NASA JOHNSON SPACE CENTER . . . . .	345
Shuttle Remote Manipulator System Mission Preparation and Operations	
E.E. Smith, Jr. . . . .	347
A Comparison of the Shuttle Remote Manipulator System and the Space Station Freedom Mobile Servicing Center	
E.C. Taylor and M. Ross . . . . .	353
Dexterous Manipulator Flight Demonstration	
E.L. Carter . . . . .	363
An Intelligent, Free-flying Robot	
G.J. Reuter, C.W. Hess, D.E. Rhoades, L.W. McFadin, K.J. Healey, J.D. Erickson, and D.E. Phinney . . . . .	373
 <u>APPENDIX A</u>	
Program Schedule . . . . .	381
 <u>APPENDIX B</u>	
Index by Author . . . . .	399
 <u>APPENDIX C</u>	
Attendees/Participants . . . . .	405



## **REDUNDANT MANIPULATORS 2**





# Characterization and Control of Self-motions in Redundant Manipulators

J. Burdick\*, H. Seraji†

## Abstract

*The presence of redundant degrees of freedom in a manipulator structure leads to a physical phenomenon known as a "self-motion," which is a continuous motion of the manipulator joints that leaves the end-effector motionless. In the first part of the paper, a global manifold mapping reformulation of manipulator kinematics is reviewed, and the inverse kinematic solution for redundant manipulators is developed in terms of self-motion manifolds. Global characterizations of the self-motion manifolds in terms of their number, geometry, homotopy class, and null space are reviewed using examples. Much previous work in redundant manipulator control has been concerned with the "redundancy resolution" problem, in which methods are developed to determine, or "resolve," the motion of the joints in order to achieve end-effector trajectory control while optimizing additional objective functions. Redundancy resolution problems can be equivalently posed as the control of self-motions. In the second part of the paper, alternatives for redundancy resolution are briefly discussed.*

## 1. Introduction

A redundant manipulator is one that has more degrees of freedom than is the minimum number nominally required to perform a given set of tasks. Redundancy in the manipulator structure yields increased dexterity and versatility for performing a task due to the infinite number of joint motions which result in the same end-effector trajectory. However, this richness of joint motions complicates the manipulator control problem considerably. In order to take advantage of redundancy, control schemes which effectively utilize redundancy in some useful manner must be developed. In recent years redundant manipulators have been the subject of considerable research, and several uses for redundancy and methods to resolve redundancy have been suggested. Much of the research on redundant manipulators has been explicitly or implicitly based on the Jacobian pseudo-inverse approach [1] for the utilization of redundancy through *local* optimization of some criterion functional.

This paper presents a different approach to the kinematics of redundant manipulators, which is based on a manifold mapping reformulation that stresses *global*, rather than *local*, kinematic analysis. Within this framework, the infinite number of redundant manipulator inverse kinematic solutions are naturally interpreted as a finite set of "self-motion manifolds." The self-motion manifold approach is a useful foundation for studying redundant manipulator kinematics. Additionally, redundancy resolution can be equivalently posed as the control of self-motions; and the self-motion manifolds are useful for investigating, interpreting, and formulating both local and global redundancy resolution techniques.

The resolution of the redundancy can be implemented by direct control of a set of self-motion parameters, by direct control of a related set of user-defined kinematic functions, or through the optimization of an objective function. Redundancy resolution can also be posed as a local or global problem.

\* Dept. of Mechanical Engineering, California Institute of Technology, Pasadena, CA

† Jet Propulsion Laboratories, California Institute of Technology, Pasadena, CA

## 2. Kinematics of Redundant Manipulators

A manipulator forward kinematic function,  $f$ , is a nonlinear vector function which relates a set of  $n$  joint coordinates,  $\theta$ , to a set of  $m$  end-effector coordinates:

$$\mathbf{x} = f(\theta). \quad (1)$$

One of the primary problems of practical interest in manipulator kinematics is determining the set of solutions to the inverse kinematic function,  $f^{-1}$ :

$$\theta = f^{-1}(\mathbf{x}). \quad (2)$$

For non-redundant manipulators, there is a finite and bounded set of joint angles which satisfy (2). Each solution corresponds to a distinct manipulator "pose." For redundant manipulators, there are an infinite number of joint angles which satisfy the inverse kinematic relation in (2), although, as will be shown, the infinity of solutions can be grouped into a finite and bounded set of smooth manifolds.

Previous redundant manipulator investigations have often focused on the linearized first order instantaneous kinematic relation between end-effector velocities and joint velocities:

$$\dot{\mathbf{x}} = \mathbf{J}(\theta)\dot{\theta} \quad (3)$$

where  $\mathbf{J}(\theta) = df(\theta)/d\theta$  is the  $m \times n$  manipulator end-effector Jacobian matrix. When  $n > m$ ,  $\mathbf{J}(\theta)$  is not uniquely invertible, and pseudo-inverse techniques [1] can be used to select a joint velocity vector, from the infinity of possible solutions, which generates a desired end-effector velocity vector. In the redundant manipulator literature, the inverse solution to (3) is often referred to as the inverse kinematic solution, rather than (2).

The redundant degrees of freedom, which lead to multiply infinite solutions in (2) and (3), can be used to perform additional tasks or optimize manipulator kinematic, dynamic, or mechanical properties. The simplest inverse solution to (3) is based on the Jacobian pseudo-inverse:

$$\dot{\theta} = \mathbf{J}_W^\dagger(\theta) \dot{\mathbf{x}} \quad (4)$$

where  $\mathbf{J}_W^\dagger(\theta) = W\mathbf{J}^T(\theta)[\mathbf{J}(\theta)W\mathbf{J}^T(\theta)]^{-1}$  is a weighted pseudo-inverse of the manipulator end-effector Jacobian matrix.  $W$  is a symmetric positive definite matrix, and the solution in (4) instantaneously minimizes the weighted quadratic form  $\dot{\theta}^T W^{-1} \dot{\theta}$  at configuration  $\theta$ . This solution can be modified by adding a null space component to the instantaneous joint velocities:

$$\dot{\theta} = \mathbf{J}_W^\dagger(\theta) \dot{\mathbf{x}} + (\mathbf{I} - \mathbf{J}_W^\dagger(\theta) \mathbf{J}(\theta))\mathbf{y} \quad (5)$$

where  $\mathbf{y}$  is an arbitrary  $n \times 1$  vector. The term  $(\mathbf{I} - \mathbf{J}_W^\dagger(\theta) \mathbf{J}(\theta))$  projects  $\mathbf{y}$  onto the null space of the manipulator Jacobian matrix. Physically, any motion in the null space is an *instantaneous* motion of the manipulator joints which causes no motion of the end-effector. Many redundancy resolution criteria can be developed as potential functions, and  $\mathbf{y}$  can be the gradient of the resolution potential function,  $g(\theta)$ :  $\mathbf{y} = \alpha \nabla g(\theta)$ , where  $\alpha$  is a scalar. Other instantaneous redundancy resolution techniques, including task optimization [9] have also been proposed.

The global inverse kinematic solution in (2) can be conveniently investigated by introducing a manifold mapping reformulation of manipulator kinematics which considers the aggregate, and thus *global*, action of the kinematic and inverse kinematic maps on the configuration space manifold. This approach allows simple topological tools to be applied to the study of manipulator kinematics [2]. The following sections present an overview of the manifold mapping

reformulation and the interpretation of redundant inverse kinematic solutions in terms of self-motion manifolds. This approach gives a natural interpretation to the solution to (2), offers useful insight into the local redundancy resolution schemes in (4) and (5), and suggests new approaches to redundancy resolution.

### 3. A Manifold Mapping Reformulation of Manipulator Kinematics

Only revolute jointed manipulators will be considered in this paper, although manipulators constructed from other lower pair joints can be similarly treated. In order to globally analyze manipulator kinematic functions it is useful to rephrase the forward and inverse kinematic problems in terms of manifold mappings. From a point-wise mapping perspective, the forward kinematic function in (1) maps a unique joint configuration,  $\theta$ , to an end-effector location,  $\mathbf{x}$ :  $\mathbf{x} = f(\theta)$ . The set of all possible joint configurations forms a space, termed the “joint space” or “configuration space,” which has a simple manifold structure. Similarly, the set of all possible end-effector locations forms the “workspace,” which also has a manifold structure.

First consider how a manipulator configuration space can be developed as a manifold. Let  $\theta_j$  denote the joint rotation angle for the  $j^{\text{th}}$  revolute joint. If the motion of the  $j^{\text{th}}$  joint is not limited due to mechanical stops,  $\theta_j$  can take on all values in the interval  $[-\pi, \pi]$ . The identification of the two end-points of the interval,  $\pi$  and  $-\pi$ , yields a circle, denoted by the symbol  $S^1$  in Figure 1. The configuration space,  $\mathcal{C}$ , of an  $n$ -revolute-jointed manipulator is a product space formed by the  $n$ -times product of the individual joint manifolds:

$$\mathcal{C} = S^1 \times S^1 \times \dots \times S^1 = T^n \quad (6)$$

where  $T^n$  is an  $n$ -torus, which is a compact  $n$ -dimensional manifold. Each of the circles that make up the torus is termed a *generator* of the torus, and is physically equivalent to a  $2\pi$  rotation of one joint. There is a one-to-one correspondence between each point in the  $n$ -torus configuration space manifold and a discrete manipulator configuration. For example, the 2R planar manipulator in Figure 2a has a 2-torus configuration space shown in Figure 2b.

While the torus geometry properly captures the topology of the configuration space manifold, there are times when other representations of the torus are useful. For example, the 2-torus representation of the 2R manipulator configuration space can be presented as a square with dimension  $2\pi$  by “cutting” the torus along two generators, as shown in Figure 3. The 3-torus configuration space of a 3R manipulator can not be directly viewed in a 3-dimensional space, but it can be presented as a cube by cutting along the 3-tori generators, as in Figure 3. These configuration space representations are also useful for plotting trajectories and surfaces in  $\theta$ -space, and all of the “cubes” in Figures 4 and 5 represent 3-tori.

To establish the geometry of the workspace, attach a frame to the manipulator end-effector. The manipulator’s workspace manifold,  $\mathcal{W}$ , is the set of all possible locations and orientations of this frame as the manipulator joints are swept through all points of the configuration space. The geometric characterization of  $\mathcal{W}$  is more complex than the torus characterization of the configuration space [2,4]. Briefly, the workspace has a “layered” or sheet-like structure.

The forward kinematic function can be viewed as a mapping of points from the configuration space to the workspace. More importantly, one can consider the action of the forward kinematic function as *the global rearrangement of the configuration space manifold to produce the workspace manifold*:

$$f(\theta): \mathcal{C} \rightarrow \mathcal{W}. \quad (7)$$

Roughly speaking, the forward kinematic map “rips” the configuration space manifold apart into pieces; distorts each piece; and combines the distorted pieces to form  $\mathcal{W}$ . A more detailed description of this mapping can be found in [2,4].

#### 4. Redundant Inverse Kinematic Solution: Self-Motion Manifolds

For non-redundant manipulators, the inverse kinematic solution (also termed a *preimage*)  $f^{-1}(\mathbf{x})$  of a regular<sup>1</sup> end-effector location is a bounded set of discrete configurations. It is known [5] that a 6R manipulator with arbitrary geometry can have up to 16 inverse kinematic solutions.

Let  $r = n - m$  be the relative degrees of redundancy. Since  $f$  is a smooth function operating on a compact manifold,  $\mathcal{C}$ ,  $f^{-1}(\mathbf{x})$  must be an  $r$ -dimensional submanifold of the configuration space [6] if  $\mathbf{x}$  is a regular value. The preimage submanifold may actually be divided into several disjoint manifolds. Formally, let a redundant inverse kinematic solution be denoted as the union of one or more disjoint  $r$ -dimensional manifolds:

$$f^{-1}(\mathbf{x}) = \bigcup_i^{n_{sm}} M_i(\psi) \quad (8)$$

where  $M_i(\psi)$  is the  $i^{\text{th}}$   $r$ -dimensional manifold in the inverse kinematic preimage and  $M_i(\psi) \cap M_j(\psi) = \emptyset$  for  $i \neq j$ . Each of the preimage manifolds can be physically interpreted as a “self-motion,” which is a continuous motion of the manipulator joints that leaves the end-effector motionless.

**Definition:** Each of the disjoint  $r$ -dimensional manifolds in the inverse kinematic preimage will be termed a *self-motion manifold*.

$n_{sm}$  is the number of self-motions in the preimage of  $\mathbf{x}$  (bounds on the value of  $n_{sm}$  will be reviewed in Section 5), and a given end-effector location may have more than one associated distinct self-motion. The multiply disjoint self-motions are akin to the distinct poses that make up non-redundant manipulator inverse kinematic solutions. Each  $M_i$  can be parametrized by a set of  $r$  independent parameters,  $\psi = \{\psi_1, \dots, \psi_r\}$ , which can be thought of as generalized coordinates for the self-motions. For a given end-effector location there is a unique choice (up to isomorphism) of self-motion parameters. However, the choice of the self-motion parameter can vary in different well-defined regions of the workspace [2]. The self-motion manifolds are best illustrated using two examples: a planar 3R manipulator (Figure 4) which is redundant with respect to the position of its end-effector, and a 4R regional manipulator which is similar to an “elbow” manipulator (Figure 5).

The self-motion manifolds of the 3R manipulator can be computed as follows. Let  $\psi$ , which is the orientation of the third link relative to a fixed reference system, be the parameter describing the internal motion of the manipulator (there are other valid, useful, and physically meaningful choices for the self-motion parameter). For a given end-effector location,  $(x_{ee}, y_{ee})$ , and an arbitrary value of  $\psi$ , there are two possible sets of joint angles,  $\{\theta_{1a}, \theta_{2a}, \theta_{3a}\}$  and  $\{\theta_{1b}, \theta_{2b}, \theta_{3b}\}$ , which can be determined by evaluating the following equations:

$$\begin{aligned} R_2 &= \sqrt{x_{ee}^2 + y_{ee}^2 + l_3^2 - 2l_3(x_{ee} \cos \psi + y_{ee} \sin \psi)} \\ \alpha &= \text{atan2}(y_{ee} - l_3 \sin \psi, x_{ee} - l_3 \cos \psi) \\ \gamma &= \cos^{-1} \left( \frac{l_1^2 + l_2^2 - R_2^2}{2l_1 l_2} \right) & \eta &= \cos^{-1} \left( \frac{l_1^2 + R_2^2 - l_2^2}{2l_1 R_2} \right) \\ \{\theta_{1a}, \theta_{2a}, \theta_{3a}\} &= \{(\alpha + \eta), (\gamma - \pi), (\psi - \alpha - \eta - \gamma + \pi)\} \\ \{\theta_{1b}, \theta_{2b}, \theta_{3b}\} &= \{(\alpha - \eta), (\pi - \gamma), (\psi - \alpha + \eta + \gamma - \pi)\} \end{aligned} \quad (9)$$

<sup>1</sup> A *regular point* of the map  $f$  is a discrete configuration,  $\theta$ , for which  $f(\theta)$  is not singular (the Jacobian of  $f$  remains full rank). A *regular value* is an end-effector location  $\mathbf{x} = f(\theta)$  where  $\theta$  is a regular point. A *critical point* is a configuration,  $\theta$ , such that  $f(\theta)$  is singular (the Jacobian of  $f$  loses rank). A *critical value* is an end-effector location  $\mathbf{x} = f(\theta)$  where  $\theta$  is a critical point.

where  $l_1, l_2, l_3$  are the lengths of links 1, 2, and 3. As  $\psi$  is swept through its feasible range of  $[-\pi, \pi]$ , equations (9) will generate two 1-dimensional manifolds in the configuration space. These manifolds may remain separate for all values of  $\psi$ , in which case there are two disjoint self-motions, or the two branches may meet at two points (corresponding to the singularities of the non-redundant  $2R$  planar manipulator subchain formed by links 1 and 2), to form one self-motion manifold. In this case, the solution in (9) becomes imaginary for some values of  $\psi$ .

Figure 4 shows the self-motion manifolds of this planar manipulator (embedded in the configuration space 3-torus) for two different locations of the end-effector. The inverse image of point 1 contains two distinct self-motion manifolds, while the inverse image of point 2 contains only one self-motion. [Note: the self-motion manifolds corresponding to point 1 are closed loops, but appear as non-closed curves because of the cubic 3-torus representation in Figure 4.] The two distinct self-motion manifolds in the preimage of point 1 physically correspond to "up elbow" and "down elbow" self-motions which are an analogous generalization of the "up elbow" and "down elbow" configurations of a non-redundant two-link manipulator. Self-motions can be thought of as a natural generalization of the non-redundant manipulator concept of "pose" to redundant manipulators. In both cases, the self-motion manifolds are diffeomorphic<sup>2</sup> to a circle. However, the preimage of point 1 contains a generator of the configuration space (a  $2\pi$  joint rotation) while the other preimage does not. These two self-motion manifolds are not homotopic<sup>3</sup>.

Now consider the more complicated  $4R$  manipulator in Figure 5. The kinematic parameters of this arm (using the modified Denavit-Hartenberg convention as in [7]) are:  $\alpha_0 = 0$ ;  $\alpha_1 = \alpha_2 = \pi/2$ ;  $\alpha_3 = -\pi/2$ ;  $a_0 = a_1 = a_2 = 0$ ;  $a_3 = a_4 = l$ ;  $d_1 = d_2 = d_3 = d_4 = 0$ . Let the self-motion parameter,  $\psi$ , be the angle between the plane containing the third and fourth links and the vertical plane passing through joint axis 1. The following equations compute four inverse kinematic solutions,  $\{\theta_{1a}, \theta_{2a}, \theta_{3a}, \theta_{4a}\}, \{\theta_{1b}, \theta_{2b}, \theta_{3b}, \theta_{4b}\}, \{\theta_{1c}, \theta_{2c}, \theta_{3c}, \theta_{4c}\}, \{\theta_{1d}, \theta_{2d}, \theta_{3d}, \theta_{4d}\}$ , given an end-effector location,  $\mathbf{x} = (x_{ee}, y_{ee}, z_{ee})$ , and a value for the self-motion parameter,  $\psi$ .

Define the following variables, which are purely functions of the end-effector location and the link length parameter,  $l$ .

$$\begin{aligned} R_2 &= \sqrt{x_{ee}^2 + y_{ee}^2}; & R_3 &= \sqrt{x_{ee}^2 + y_{ee}^2 + z_{ee}^2}. \\ \cos \beta &= x_{ee}/R_2; & \sin \beta &= y_{ee}/R_2 & \beta &= \text{atan2}(y_{ee}, x_{ee}). \\ \cos \xi &= R_2/R_3; & \sin \xi &= z_{ee}/R_3; & \xi &= \text{atan2}(z_{ee}, R_2). \\ \cos \gamma &= R_3/2l; & \sin \gamma &= (1/2l)\sqrt{4l^2 - R_3^2}; & \gamma &= \text{atan2}(\sin \gamma, \cos \gamma). \end{aligned} \quad (10)$$

There are two unique values of  $\theta_4$  which satisfy the inverse kinematic function:

$$\theta_{4a} = 2\gamma; \quad \theta_{4b} = -\theta_{4a} \quad (11)$$

<sup>2</sup> A smooth map  $f: X \rightarrow Y$  (where  $X$  and  $Y$  are manifolds) is a diffeomorphism if it is one-to-one and onto, and  $f^{-1}: Y \rightarrow X$  is smooth.  $X$  and  $Y$  are diffeomorphic if such an  $f$  exists.

<sup>3</sup> Two maps,  $f_0: X \rightarrow Y$  and  $f_1: X \rightarrow Y$  are homotopic if there exists a smooth map,  $F: X \times I \rightarrow Y$  such that  $F(x, 0) = f_0(x)$  and  $F(x, 1) = f_1(x)$ . In other words,  $f_0$  can be deformed to  $f_1$  through a smoothly evolving family of maps, and two self-motion manifolds are homotopic if one can be continuously deformed into the other continuously on the surface of the configuration space torus.

There are four unique values of  $\theta_2$  (two corresponding to each value of  $\theta_4$  in (12)):

$$\begin{aligned}\theta_{2a} &= \cos^{-1}[\sin \xi \sin \gamma - \cos \xi \cos \gamma \cos \psi] & \theta_{2b} &= -\theta_{2a} \\ \theta_{2c} &= \cos^{-1}[\sin \xi \sin \gamma + \cos \xi \cos \gamma \cos \psi] & \theta_{2d} &= -\theta_{2c}.\end{aligned}\quad (12)$$

There are four corresponding values of  $\theta_1$  which can be computed as follows:

$$\begin{aligned}\theta_{1a} &= \text{atan2} \left[ \frac{-\sin \psi \cos \gamma}{\sin \theta_{2a}}, \frac{-(\cos \xi \sin \gamma + \sin \xi \cos \gamma \cos \psi)}{\sin \theta_{2a}} \right] & \theta_{1b} &= \theta_{1a} \pm \pi \\ \theta_{1c} &= \text{atan2} \left[ \frac{\sin \psi \cos \gamma}{\sin \theta_{2c}}, \frac{(-\cos \xi \sin \gamma + \sin \xi \cos \gamma \cos \psi)}{\sin \theta_{2c}} \right] & \theta_{1d} &= \theta_{1c} \pm \pi\end{aligned}\quad (13)$$

Similarly, there are four corresponding values of  $\theta_3$  which can be computed as follows:

$$\begin{aligned}\theta_{3a} &= \text{atan2} \left[ \frac{-\cos \xi \sin \psi}{\sin \theta_{2a}}, \frac{\sin \xi \cos \gamma + \cos \xi \sin \gamma \cos \psi}{\sin \theta_{2a}} \right] & \theta_{3b} &= \theta_{3a} \pm \pi \\ \theta_{3c} &= \text{atan2} \left[ \frac{\cos \xi \sin \psi}{\sin \theta_{2c}}, \frac{\sin \xi \cos \gamma + \cos \xi \sin \gamma \cos \psi}{\sin \theta_{2c}} \right] & \theta_{3d} &= \theta_{3c} \pm \pi\end{aligned}\quad (14)$$

The inverse solution is real for all values of  $\psi$  in the range  $[-\pi, \pi]$ . The four distinct self-motions of this manipulator can be generated by continuously sweeping  $\psi$  through its  $2\pi$  range for fixed  $(x_{ee}, y_{ee}, z_{ee})$ . Figure 6 shows the cubic representation of the projection of these four self-motions onto the  $\theta_1$ - $\theta_2$ - $\theta_3$  and  $\theta_2$ - $\theta_3$ - $\theta_4$  3-tori (for the case in which  $l = 1.0$ ,  $(x_{ee}, y_{ee}, z_{ee}) = (0.0, 1.0, 0.9)$ ).

## 5. Characterizations of the Self-Motion Manifolds and the Jacobian Null Space

A more detailed study of the number, geometry, and homotopy classes of self-motion manifolds can be found in [2,3]. Some of the relevant results are requested here.

**Theorem 1:** *An  $n$ -revolute-jointed redundant manipulator can have no more self-motions than the maximum number of inverse kinematic solutions of a non-redundant manipulator of the same class. That is, for a fixed end-effector location, redundant spherical, regional, and spatial manipulators with an arbitrary number of revolute joints can respectively have as many as 2, 4, and 16 distinct self-motions.*

**Theorem 2:** *The self-motion manifolds of an  $n$ -revolute-jointed redundant manipulator are diffeomorphic to  $T^r$ , an  $r$ -dimensional torus.*

Theorem 1 says that manipulators with arbitrary geometry have an upper bound on the number of self-motions for a fixed end-effector location. Roughly speaking, Theorem 2 says that each self-motion manifold is a distorted  $r$ -dimensional torus lying in the  $n$ -torus configuration space. This result actually holds for non-redundant manipulators as well, since the inverse image must be a 0-dimensional torus, or a point.

Self motions which are homotopic to each other form a *homotopy class*. The notion of a homotopy class has previously been used in [8] to characterize different redundancy resolution paths. While an exact bound on the possible number of self-motion homotopy classes which exist for a given manipulator has not been rigorously determined:

**Proposition 3:** *An  $n$ -revolute-jointed ( $n \geq 7$ ) spatial manipulator can have as many as  $2^{(n-2)}$  different self-motion homotopy classes.*

Many redundancy resolution techniques employ the null space of the manipulator Jacobian. In [3] it is shown that the null space has a simple interpretation as the self-motion manifold tangent space.

**Theorem 4:** The null space of the Jacobian matrix, evaluated at a particular joint configuration,  $\theta_o$ , is the tangent to the self-motion manifold at  $\theta_o$ .

Theorem 4 is particularly useful for interpreting instantaneous redundancy resolution techniques, such as (4) and (5).

## 6. User-Defined Kinematic Functions and the Augmented Jacobian

Locally, the  $m$  end-effector coordinates and the  $r$  self-motion parameters constitute a set of generalized coordinates for a redundant manipulator. Due to the multiplicity of self-motions, there are multiple sets of these generalized coordinates in different subregions of the configuration space. While  $\mathbf{x}$  and  $\psi$  are a valid set of generalized coordinates for control of the redundant manipulator,  $\psi$  is not always physically meaningful or the direct parameters of interest in performing a task.

It is often expedient to define  $r$  kinematic functions  $\phi = \{\phi_1(\theta), \phi_2(\theta), \dots, \phi_r(\theta)\}$  to reflect some "additional task" that will be performed with the manipulator redundancy. Each  $\phi_i$  can be a function of the joint angles  $\{\theta_1, \dots, \theta_n\}$  and the link geometric parameters. The user-defined kinematic functions can, for example, be the coordinates of a point on the manipulator, or the angle of a link with respect to a fixed reference system. However, for the set  $\{\mathbf{x}, \phi\}$  to constitute a proper set of generalized coordinates, the user-defined kinematic functions must be expressible as independent functions of  $\psi$ :  $\phi = \{\phi_1(\theta(\psi)), \dots, \phi_r(\theta(\psi))\}$ .

Let us consider an augmented task vector comprised of the end-effector coordinates and the self-motion parameters,  $\{\mathbf{x}, \psi\}$ . This new set of generalized coordinates can be instantaneously related to the manipulator joint angles through the *basic augmented Jacobian*:

$$\begin{bmatrix} \mathbf{J} \\ \mathbf{J}_\psi \end{bmatrix} = \begin{bmatrix} d\mathbf{x}/d\theta \\ d\psi/d\theta \end{bmatrix}. \quad (15)$$

A set of generalized coordinates based on the end-effector coordinates and the user-defined kinematic function,  $\{\mathbf{x}, \phi(\psi)\}$ , can be instantaneously related to the joint coordinates through the *augmented Jacobian* of the form:

$$\begin{bmatrix} \dot{\mathbf{x}} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \frac{d\phi}{d\psi} \\ 0 & \frac{d\psi}{d\theta} \end{bmatrix} \begin{bmatrix} \mathbf{J} \\ \mathbf{J}_\psi \end{bmatrix} \dot{\theta}. \quad (16)$$

The augmented Jacobian will be singular whenever the basic augmented Jacobian in (15) loses rank and/or when the matrix  $d\phi/d\psi$  loses rank. Let us focus on the singularities of the basic augmented Jacobian since they are invariant to the particular selection of user-defined kinematic functions. A singularity of the augmented Jacobian will also cause the following matrix to lose rank:

$$\begin{bmatrix} \mathbf{J} \\ \mathbf{J}_\psi \end{bmatrix} \begin{bmatrix} \mathbf{J}^T & \mathbf{J}_\psi^T \end{bmatrix} = \begin{bmatrix} \mathbf{J}\mathbf{J}^T & \mathbf{J}\mathbf{J}_\psi^T \\ \mathbf{J}_\psi\mathbf{J}^T & \mathbf{J}_\psi\mathbf{J}_\psi^T \end{bmatrix}. \quad (17)$$

Since the matrix in (17) is square, a deficiency in its rank can be determined by investigating its determinant:

$$\det \begin{bmatrix} \mathbf{J}\mathbf{J}^T & \mathbf{J}\mathbf{J}_\psi^T \\ \mathbf{J}_\psi\mathbf{J}^T & \mathbf{J}_\psi\mathbf{J}_\psi^T \end{bmatrix} = \det(\mathbf{J}\mathbf{J}^T) \cdot \det[\mathbf{J}_\psi(\mathbf{I} - \mathbf{J}^\dagger \mathbf{J})\mathbf{J}_\psi^T] \quad (18)$$

There are three conditions which lead to zero determinant in (18). The basic augmented Jacobian will lose rank when  $\mathbf{J}$  loses rank (condition 1). The augmented Jacobian will also lose rank when  $\det[\mathbf{J}_\psi(\mathbf{I} - \mathbf{J}^\dagger \mathbf{J})\mathbf{J}_\psi^T] = 0$ . This will occur when  $\mathbf{J}_\psi$  is rank-deficient (condition 2).

Since  $(\mathbf{I} - \mathbf{J}^\dagger \mathbf{J})$  is the null space projection operator, the augmented Jacobian will also lose rank when one or more rows of  $\mathbf{J}_\psi$  are orthogonal to the null space of  $\mathbf{J}$ . If the rows of  $\mathbf{J}_\psi$  are orthogonal to the null space of  $\mathbf{J}$ , the null space of  $\mathbf{J}$  is in the null space of  $\mathbf{J}_\psi$ . Therefore, the augmented Jacobian will also lose rank when  $N(\mathbf{J}) \cap N(\mathbf{J}_\psi) \neq \emptyset$ , where  $N(\mathbf{J})$  and  $N(\mathbf{J}_\psi)$  are the null spaces of  $\mathbf{J}$  and  $\mathbf{J}_\psi$  (condition 3).

## 7. Alternatives for Redundancy Resolution

There are many alternative ways to implement redundancy resolution. Redundancy resolution can be effected by direct trajectory control of the manipulator generalized coordinates,  $\{\mathbf{x}(t), \psi(t)\}$ , or the related coordinates,  $\{\mathbf{x}(t), \phi(t)\}$ , throughout the motion. This approach can be used to achieve a desirable evolution of the robot configuration while working in a confined space, or to avoid workspace obstacles, kinematic singularities, or joint limits [10]. An adaptive configuration control scheme, such as [10], or an extension of the operational space method [11] can be used to directly control the manipulator in the task coordinates. The method outlined in [10] also permits inequality constraints on the user-defined functions,  $\phi(t)$ . Alternatively, the equivalent joint space trajectories can be computed as a function of the specified end-effector and self-motion trajectories:  $\theta(t) = f^{-1}(\mathbf{x}(t), \psi(t))$ , and joint based control can be used to servo the manipulator along the joint trajectories. Equations (9) and (10-14) are examples of inverse kinematic functions for redundant manipulators.

In another approach, redundancy can be used to optimize a desirable objective function. Let  $g(\theta)$  denote a scalar objective function to be optimized with the redundant degrees of freedom. The criterion for optimizing  $g(\theta)$  with the constraint  $\dot{\mathbf{x}} = \mathbf{J}\dot{\theta}$  is:

$$B \frac{\partial g}{\partial \theta} = 0 \quad (19)$$

where  $B$  is an  $r \times n$  matrix formed from  $r$  linearly independent rows of  $(\mathbf{I} - \mathbf{J}^\dagger \mathbf{J})$ . Equation (19) is the result used by Baillieul [9] in the Extended Jacobian Method. Other methods for optimizing  $g(\theta)$  generally lead to Jacobian based methods, such as (4) and (5). However, by defining  $\phi(\theta) = B \partial g / \partial \theta$ , we can see that kinematic optimization can also be reformulated as a special case of trajectory tracking.

Redundancy resolution problems can be posed as either local or global problems. In local approaches, such as (4) and (5), the objective function is *instantaneously* optimized. In global approaches, a global measure of the objective function is optimized over the length of a trajectory.

Local methods suffer from two drawbacks, which can be illustrated by considering a redundancy resolution problem with fixed end-effector location. The goal is to find the configuration which maximizes an objective function. Assume that the manipulator starts with the end-effector at the proper location, but with a configuration which does not maximize the objective function. In the null space based approach of (5), an incremental change in joint angle position (which will move the manipulator closer to the optimal configuration) is computed by projecting the gradient of the objective function onto the null space. This operation is equivalent to a first order gradient search over the self-motion manifold. First order gradient searches are prone to finding *local*, rather than global, maximum. Second, the null space techniques can only optimize over a single self-motion manifold. The true optimal configuration might be contained in another disjoint self-motion manifold, and consequently can not be found by such a technique.



The concept of a self-motion manifold can be useful for developing other approaches to global redundancy resolution. Again consider the case of redundancy resolution for fixed end-effector location. Let  $g(\theta)$  be the objective function. For fixed end-effector location,  $\theta$  is implicitly a function of  $\psi$ . Global redundancy resolution is then equivalent to finding:

$$\max_i \left( \max_{\psi} g_i(\psi) \right) \quad (i = 1, \dots, n_{sm}) \quad (20)$$

where  $g_i(\psi)$  is the restriction of  $g(\psi)$  to the  $i^{\text{th}}$  self-motion manifold. The local maxima and minima of the resolution function can be found as the roots to  $n_{sm}$  polynomials:

$$\frac{dg_i(\psi)}{d\psi} = 0 \quad (i = 1, \dots, n_{sm}). \quad (21)$$

the globally optimal solution is selected by evaluating  $g(\psi)$  at each of the locally optimal roots to equations (21). Once the optimal set of self-motion parameters has been determined, the corresponding configuration,  $\theta_{\text{optimal}}$ , can be computed from the inverse kinematic function. To extend this approach to redundancy resolution along a trajectory, the polynomial equations could be solved along the path, and the multiple trajectories compared to determine the globally optimal one.

## 9. Conclusions

Self-motions are inherent in redundant manipulators, and understanding their characterization and control is important for establishing useful redundancy resolution techniques. This paper reviews a global manifold mapping reformulation of manipulator kinematics in which self-motions are naturally interpreted as sub-manifolds of the configuration space. Ways to characterize these self-motion manifolds are presented. Redundancy resolution techniques can be naturally reformulated and reinterpreted in terms of self-motion manifolds, rather than the Jacobian null space. This approach yields useful global insight into redundancy resolution.

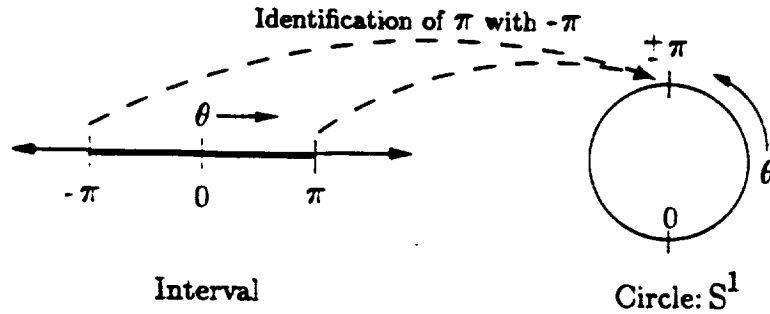
## 9. Acknowledgements

The first author gratefully acknowledges the support of the System Development Foundation for part of this work, and many helpful comments from Professor Bernard Roth, Dr. Madhu Raghavan, and Dr. Charles Wampler. This research was performed in part at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration.

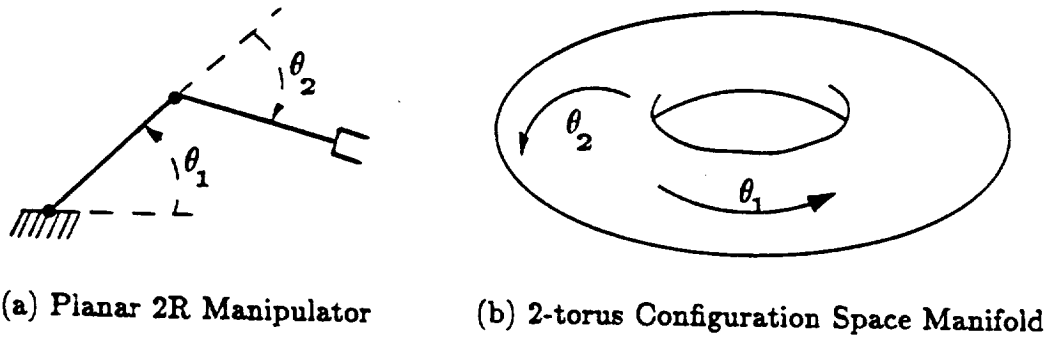
## 10. References

- [1] C.A. Klein and C.H. Huang, "Review of the Pseudoinverse for Control of Kinematically Redundant Manipulators," *IEEE Transactions on Systems, Man and Cybernetics*, March, 1983.
- [2] J. W. Burdick, "Kinematic Analysis and Design of Redundant Robot Manipulators," Ph.D. Thesis, Department of Mechanical Engineering, Stanford University, March 1988. To be issued as Stanford C.S. Report No. STAN-CS-88-1207.
- [3] J. W. Burdick, "On the Inverse Kinematics of Redundant Manipulators: Characterization of the Self-Motion Manifolds," to appear in the proceedings of the *IEEE Robotics and Automation Conference*, Scottsdale, AZ, May, 1989.
- [4] J. W. Burdick, "On the Kinematics of Redundant Manipulators: a Global Mapping Approach," in preparation.

- [5] E.J.F. Primrose, "On the Input-Output Equations of the General 7R-Mechanism," *Mechanism and Machine Theory*, Vol. 21, No. 6, pp. 509-510, 1986.
- [6] V. Guillemin and A. Pollack, *Differential Topology*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1974.
- [7] J.J. Craig, *Introduction to Robotics: Manipulation and Control*, Addison-Wesley, Reading, Mass, 1986.
- [8] J. Baillieul and D.P. Martin, "Issues in the Control of Kinematically Redundant Mechanisms," *Proceedings of the NATO Workshop on Redundancy in Robotics: Sensing, Design, and Control*, Salo, Italy, June 26-July 1, 1988, Springer Verlag.
- [9] J. Baillieul, "Kinematic Programming Alternatives for Redundant Manipulators," *Proceedings of the IEEE International Conference on Robotics and Automation*, St. Louis, MO, March 25-28, 1985, pp. 722-728.
- [10] H. Seraji, "Configuration Control of Redundant Manipulators," *Proceedings of the NATO Advanced Research Workshop on Robots with Redundancy: Design, Sensing, and Control*, Salo, Italy, June 26-July 1, 1988, Springer-Verlag.
- [11] O. Khatib, "Dynamic Control of Manipulators in Operational Space," *Sixth CISM-IFTOMM Congress on the Theory of Mechanisms and Machines*, New Delhi, India, Dec. 15-20, 1983, pp. 1128-1131.



**Figure 1:** Revolute Joint Configuration Space Manifold



**Figure 2:** Configuration Space of a 2R Manipulator

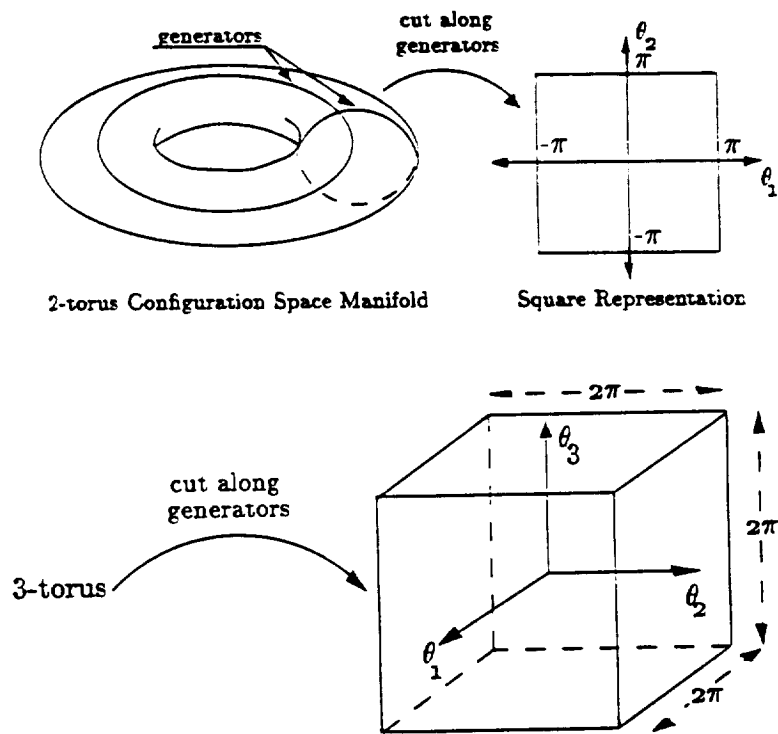


Figure 3: Square and Cube Representation of a 2-torus and 3-torus

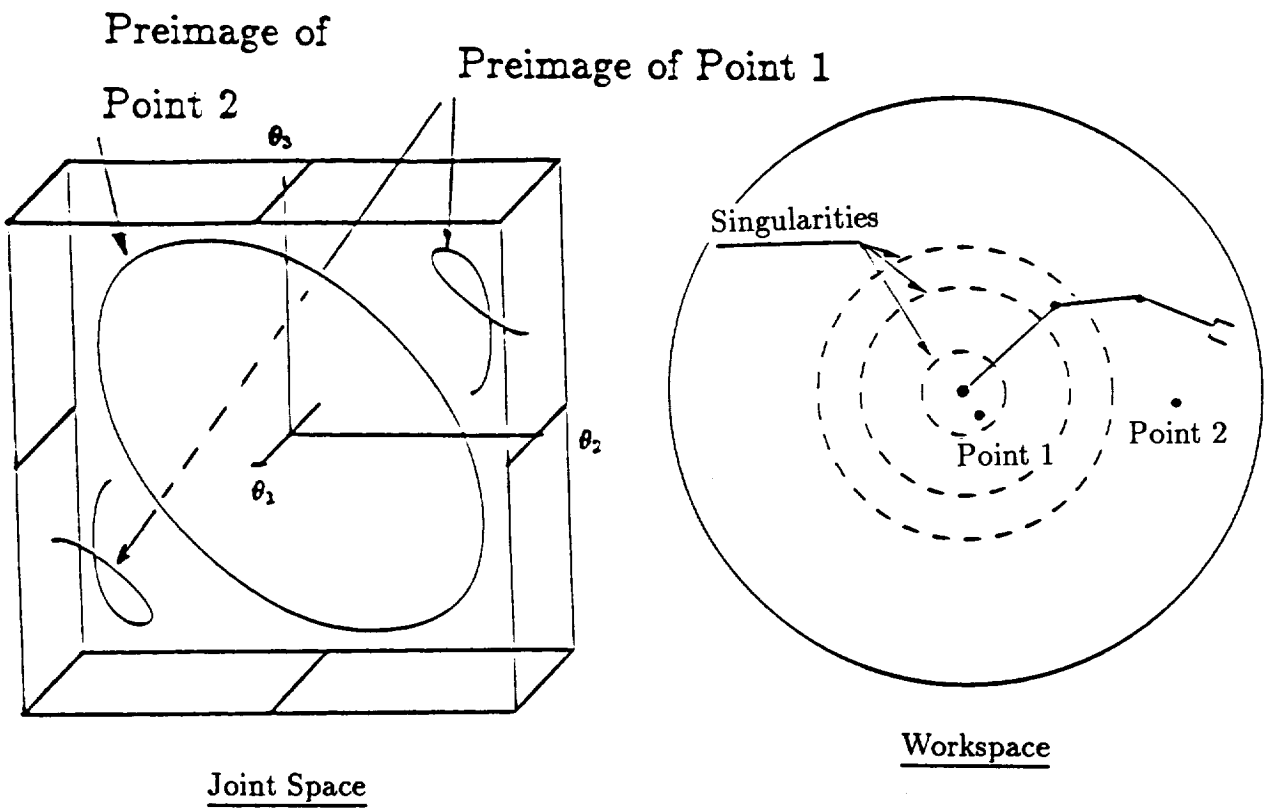


Figure 4: Self-Motion Manifolds for Two Regular Values in the Workspace of a 3R Planar Manipulator

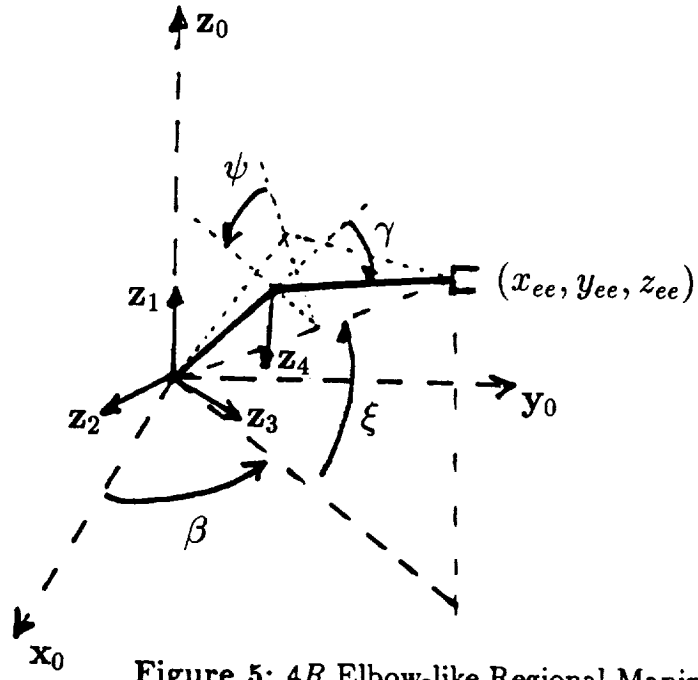
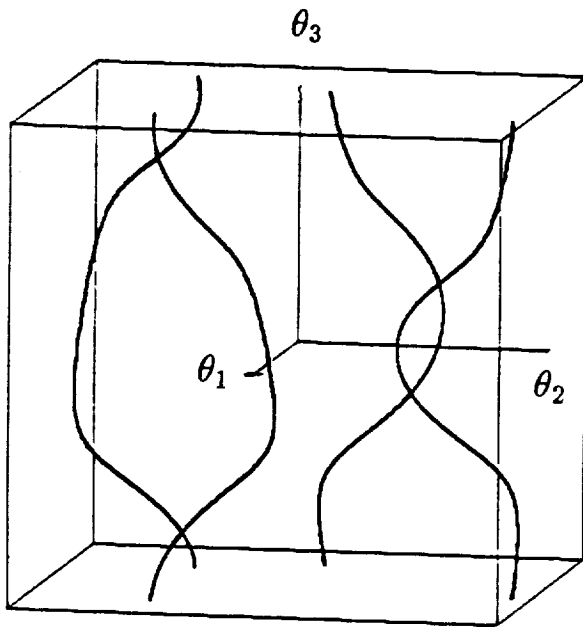
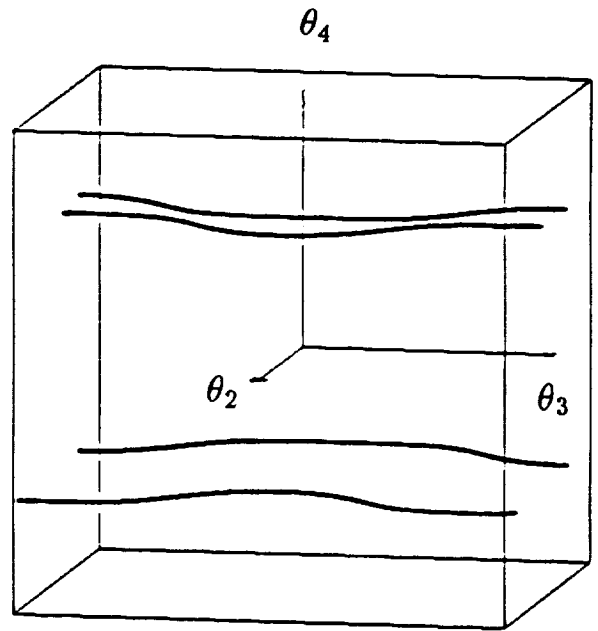


Figure 5: 4R Elbow-like Regional Manipulator



Projection onto  $\theta_1$ - $\theta_2$ - $\theta_3$  torus



Projection onto  $\theta_2$ - $\theta_3$ - $\theta_4$  torus

Figure 6: 4R Elbow-like Manipulator Self-Motion Manifolds

# MULTIPLE COOPERATING MANIPULATORS: THE CASE OF KINEMATICALLY REDUNDANT ARMS

Ian D. Walker†, Robert A. Freeman‡, and Steven I. Marcus†

†Department of Electrical and Computer Engineering

‡Department of Mechanical Engineering

The University of Texas at Austin

Austin, Texas 78712

## Abstract

This paper continues and extends existing work concerning two or more manipulators simultaneously grasping and transferring a common load. We specifically consider the case of one or more arms being kinematically redundant. Some existing results in the modeling and control of single redundant arms and multiple manipulators are reviewed. The cooperating situation is modeled in terms of a set of coordinates representing object motion and internal object squeezing. Nominal trajectories in these coordinates are produced via actuator load distribution algorithms introduced previously. A controller is developed to track these desired object trajectories while making use of the kinematic redundancy to additionally aid the cooperation and coordination of the system. It is shown how the existence of kinematic redundancy within the system may be used to enhance the degree of cooperation achievable.

## 1. INTRODUCTION

Recently, there has been increasing interest and research on the subject of multiple manipulators cooperating on a single task. This seems a natural extension of the notion of single manipulators performing tasks, and interest stems in no small part from NASA's declared interest in such systems for servicing the planned space station [12] and other extraterrestrial tasks.

Indeed, it may be in space that practical coordinated multi-arm systems find their first useful deployment. NASA has already identified a number of areas in which dual or multi-arm systems may be exploited in the space station, including servicing, assembly and construction, and launch, retrieval and handling of payloads [27, p.34]. In this paper, we consider the loading and control problems of such systems, to include both nonredundant and redundant arms.

Although the first space multi-arm systems will probably be nonredundant, it is felt that the current interest and research in single redundant arms (for some examples, see [4], [15], [23]) should, in a reasonably short time period, make the natural advantages of redundant arms (increased maneuverability, obstacle avoidance, subtask performance, etc.) available. It is natural to expect these systems to be exploited in the types of space-based applications mentioned above.

Some initial analysis of redundant arms in cooperating systems has already been made [8]-[10], [16], [21]. Much other work in various aspects of multi-arm systems has appeared recently, in modeling [5], [6], [15], [19], load distribution [20]-[22], [26], grasping [3], [13], [25], and control [1], [2], [7]-[9], [11], [14], [17], [18], [24]. In this paper we consider the case of  $L$  cooperating arms. In particular, we extend the work of Uchiyama and Dauchez [18], which considers two nonredundant arms, to  $L$  (possibly redundant) arms,

---

† This research was supported in part by the Air Force Office of Scientific Research under Grant AFOSR-86-0029, in part by the National Science Foundation under Grant ECS-8617860, and in part by the DoD Joint Services Electronics Program through the Air Force Office of Scientific Research (AFSC) Contract F49620-86-C-0045.

‡ This research was supported in part by the Bureau of Engineering Research, Contract #30-4220-4950.

employing coordinates specifying motion of the commonly held object and internal squeezing forces, which cause no motion but contribute to internal stresses in the system. The case of redundant arms is included by extending these coordinates.

A control structure is developed using the method of feedback linearization, to control the system in terms of the coordinates previously developed. This is similar in spirit to the work of Kreutz and Lokshin [9], Li, Hsu and Sastry [11], and Wen and Kreutz [24]. However, in contrast to [9] and [24], internal forces may be controlled and are represented directly at the common object coordinate level, as opposed to [11], where they are represented at the end effector level.

The paper is organized as follows. Preliminaries and some results on load distribution are presented in Section 2. Section 3 contains the necessary coordinate transformations and control structure in the nonredundant case, and this is extended to include redundant arms in Section 4. Conclusions are presented in Section 5.

## 2. LOAD DISTRIBUTION

When two (or more) manipulators form a closed chain (or chains) in cooperatively grasping a common object, the mobility of the resulting system will usually be lower than the total number of actuators available in the arms. This overabundance of kinematically dependent inputs arises from the kinematic constraints necessary to maintain the closed chain(s). If it is desired to utilize  $R$  actuators, where  $R$  is greater than the system mobility, then the loading in the system is not uniquely specified by a trajectory of the commonly grasped object. This means that in controlling such systems, a choice must be made, either implicitly or explicitly, to select one from a number of possible loading states at each point along the trajectory.

Clearly, since different selections will give rise to different internal loadings and actuator demands, some loading states will be more desirable than others — indeed, some may be unacceptable in practice, requiring unachievable actuator loads or imposing unsustainable internal forces on the manipulators and/or commonly held object. Investigation of these issues has resulted in a number of suggested solutions of the load distribution problem ([1], [7]-[9], [13], [14], [16], [18], [20], [21], [24]-[26]). In these, the common emphasis is on specifying end-effector forces/moments throughout the trajectory, subject to criteria involving internal object forces or joint torque requirements.

For this, the motion equations for the commonly held object are developed using the Newton and Euler equations

$$\underline{f} = m\ddot{\underline{r}} \quad (2.1)$$

$$\underline{n} = I\dot{\underline{\omega}} + \underline{\omega} \times (I\underline{\omega}) \quad (2.2)$$

where  $m$  and  $I$  are the mass and inertia matrix of the object, and  $\underline{r}$  and  $\underline{\omega}$  are the position of the center of mass of the object (expressed in a global reference frame) and the angular velocity of the object, respectively.

Assuming the rigid object is rigidly grasped by  $L$  robotic arms (or fingers), and that the arms may impart forces and moments to the object,  $\underline{f}$  and  $\underline{n}$  are given by

$$\underline{f} = \sum_{i=1}^L \underline{f}_i + m\underline{g} \quad (2.3)$$

$$\underline{n} = \sum_{i=1}^L \underline{n}_i + \sum_{i=1}^L \underline{s}_i \times \underline{f}_i \quad (2.4)$$

where  $\underline{f}_i$ ,  $\underline{n}_i$  are the forces and moments applied to the object by the  $i^{th}$  end effector, respectively,  $\underline{g}$  is the gravity vector, and  $\underline{s}_i$  is the position of the  $i^{th}$  end effector with respect to the object coordinates associated with  $\underline{f}$  and  $\underline{n}$  (see Fig. 1).

Equations (2.1)-(2.4) may be combined to yield

$$\underline{P} = W \underline{F} \quad (2.5)$$

where

$$\underline{P} = R \ddot{\underline{\phi}} + \underline{Q}$$

is the generalized load required to move the object, with

$$R = \begin{bmatrix} mI_u & 0 \\ 0 & I \end{bmatrix}$$

$$\ddot{\underline{\phi}} = [\ddot{\underline{r}}^T \quad \ddot{\underline{\omega}}^T]^T$$

$$\underline{Q} = \begin{bmatrix} -m\underline{g} \\ \underline{\omega} \times (I\underline{\omega}) \end{bmatrix}$$

and the loads being imparted by the  $L$  arms are

$$\underline{F} = [\underline{f}_1^T, \underline{n}_1^T, \dots, \underline{f}_L^T, \underline{n}_L^T]^T$$

with

$$W = \begin{bmatrix} I_v & 0 & \dots & I_v & 0 \\ S_1 & I_v & \dots & S_L & I_v \end{bmatrix}$$

$$S_i = \begin{bmatrix} 0 & -S_{i3} & S_{i2} \\ S_{i3} & 0 & -S_{i1} \\ -S_{i2} & S_{i1} & 0 \end{bmatrix}$$

$$\underline{s}_i = [S_{i1} \quad S_{i2} \quad S_{i3}]^T$$

$I_v$  is the  $v \times v$  identity matrix (the dimension of  $v$  is determined by the particular problem being solved, e.g.  $u = v = 3$  in the spatial case).

We have multiple solutions in general for the end effector forces  $\underline{F}$ , given the desired object motion represented by  $\underline{P}$ . The general solution to (2.5) is given by

$$\underline{F} = W^+ \underline{P} + (I_l - W^+ W) \underline{\epsilon} \quad (2.6)$$

where  $W^+ \in \mathbb{R}^{l \times k}$  (where  $k$  and  $l$  are the dimensions of  $\underline{P}$  and  $\underline{F}$  respectively) is a generalized inverse, or pseudoinverse of  $W$  and may be defined in general by  $W^+ = A W^T (W A W^T)^{-1}$  for some positive definite  $A$ . In (2.6),  $\underline{\epsilon} \in \mathbb{R}^l$  is an arbitrary vector whose choice determines which of the possible solutions of (2.5) is chosen.

Notice that choosing an  $\underline{\epsilon}$  in (2.6) represents one method of load distribution, from the object load  $\underline{P}$  to end effector loads  $\underline{F}$ , which for nonredundant arms uniquely specifies the joint torques (the redundant case is discussed in Section 4). Various authors, e.g. [13], [25], have suggested schemes to choose  $\underline{\epsilon}$  in (2.6) subject to object-related criteria, such as 'squeezing' effects on the object. However, care must be taken in such approaches to allow for the inherent squeezing effects in (2.6) due to the first term  $W^+ \underline{P}$ . The contributions to internal forces by this term (which include unavoidable inertial loadings) are investigated and quantified in [22].

Alternatively, by looking at the effects  $\underline{F}_i$  of end effector forces  $\underline{F}_i$  at the object coordinates (where  $\underline{F} = [\underline{F}_1^T, \dots, \underline{F}_L^T]^T$  and  $\underline{F}_i = [f_i^T, n_i^T]^T$ ), we have

$$\underline{P} = [I_1 \quad \dots \quad I_L] \underline{F} := \overline{W} \underline{F} \quad (2.7)$$

which is clearly similar in form to (2.5). At these coordinates, an equation similar to (2.6) may be formed, and load distribution made at this level. This is the impetus of several approaches ([1], [7], [8], [14], [18], [26]).

The appropriate transformation in obtaining (2.7) is

$$\bar{F}_i = \begin{bmatrix} I & 0 \\ S_i & I \end{bmatrix} F_i = W_i F_i, \quad (2.8)$$

the use of which will be exploited further in Section 3.

A different strategy for load distribution is to consider the manipulator dynamics of the cooperating system, which may be expressed as (see [5], [6], [21], for full details of the notation used here)

$$\tau_\beta = [M(\beta)]\ddot{\beta} + \dot{\beta}^T [N(\beta)]\dot{\beta} - J_\beta^T(\beta)F \quad (2.9)$$

(where  $\beta$  represents the joint variables of the  $L$  manipulators,  $M$  is the effective inertia matrix,  $N$  involves the centripetal and coriolis effects,  $J_\beta$  is the Jacobian relating end effector variables to joint variables  $\beta$  and  $\tau_\beta$  is the force/torque control vector corresponding to  $\beta$ ).

Consideration of load distribution effects at this (joint) level allows actuator loading effects to be taken into account, in contrast to the purely object-based criteria mentioned above. Work in this area has been performed by the authors [20], [21] using (2.6) combined with (2.9) to use joint torque based criteria to select  $\underline{\epsilon}$ , hence loading throughout the system. Joint space criteria are also used in [8], [14], [26], in which (2.7) is utilized in place of (2.6).

However, for such joint-based methods, it is necessary to monitor internal forces built up in the object as a result of the distribution strategy applied. Motion which alleviates actuator demands will not be successful if internal forces built up destroy the object transported (of course the opposite logic applies to object based methods, where end effector forces calculated must be attainable by realistic actuator torque levels). The work in [8], [26] utilizes joint space criteria subject to

$$\bar{F}_i = \alpha_i \underline{P} \quad (2.10)$$

where  $\sum_{i=1}^L \alpha_i = 1$ , and each  $\alpha_i$  is a non-negative scalar. This approach apportions only motion-causing components to  $\bar{F}$ ; thus, only the unavoidable inertial forces referred to earlier will build up in the object. However, this excludes examples such as in the two-arm case,  $\alpha_1 = -1$  and  $\alpha_2 = 2$ , which will also produce the desired object motion  $\underline{P}$ . In other words, (2.10) requires all arms to be ‘pulling in the same direction.’ This may not always be desirable for the individual arms, and indeed, in some cases it may be desirable to have forces propagated through the object (for example, to allow one arm to assist in the motion of another in certain configurations where motion in a given direction is more readily obtained by the actuator load state of that arm — notice again that modeling this situation is inherently joint-based).

Notice also that, again for the two-arm case, we have two distinct situations: (a)  $\{0 \leq \alpha_1 \leq 1 \text{ and } \alpha_2 = 1 - \alpha_1\}$ ; or (b)  $\{(\alpha_1 < 0, \alpha_2 = 1 - \alpha_1) \text{ or } (\alpha_2 < 0, \alpha_1 = 1 - \alpha_2)\}$ , which correspond to ‘pulling together’ or ‘squeezing,’ as discussed above. In both cases the arms ‘cooperate’ in the sense that the desired object motion is obtained, but only the first case represents load sharing in the sense of (2.10). Such issues should be considered in those cases, such as [14], when bias squeezing effects are added to solutions developed on the basis of (2.10).

In the case of the methods of [20], [21], object loading again needs to be regulated. This issue is addressed in [22], and the case of combined joint/object space distribution methods is the subject of current research. In the following section we consider the problem of control, and propose a methodology for controlling object motion and internal object squeezing applicable to the types of load distribution discussed above. This is extended to the case of redundant arms in Section 4.



### 3. OBJECT COORDINATE CONTROL

For the case of multiple manipulators, various control schemes have already been proposed of the feedback linearization, or 'computed torque,' type. Various philosophies of control synthesis have been proposed, including the use of LQ theory [17] and various PD or PID strategies. Of the second type, methods differ mainly in the types of coordinates to be controlled, from joint coordinates [14], [24] to end effector or operational space coordinates [8], to generalized object coordinates [3], [7], [9], [11]. In this section, we propose a controller of the computed torque type based on coordinates related directly to object motion and squeezing.

The goal is to obtain a control structure at the object level in which various strategies may be implemented. The coordinate system established here allows direct access to the force and motion coordinates involved in internal object squeezing. It is felt that nominal trajectories in these coordinates will, in many cases, provide the most clear description of the cooperating arm situation. For example, depending on the held object and task, a particular internal force strategy (perhaps no internal squeezing in given directions) may be mandatory, and this type of situation is clearly described in terms of the coordinate model introduced below.

Alternatively, nominal trajectories may be established by various other means. For example, load distribution algorithms of the type described in the previous section result in desired loadings for  $\underline{\tau}$ ,  $\underline{F}$ , or  $\underline{\bar{F}}$ , and this is easily converted to desired loadings in our object coordinates (for a discussion of this, and examples of its importance in load distribution methods, see [22]). The control structure developed in this section allows the adoption of the above, and other types of methodologies, in a straightforward approach to object space control.

First, we must establish the relevant coordinate transformations necessary. The coordinates to be controlled are the object motion coordinates  $\underline{\phi}$  in (2.5), together with coordinates  $\Delta \underline{z}$  to be defined, where  $\Delta \underline{z} \in \mathbb{R}^{6(L-1)}$  in general, and represent relative small displacements between end effector motions referenced at the generalized object coordinates. For the case of two nonredundant arms, the transformations have been developed by Uchiyama and Dauchez [18] who applied a hybrid position/force scheme to the result. We extend the result here to cover  $L$  arms.

We have seen in (2.8) how end effector forces/moments  $\underline{F}_i$  may be related to their equivalent effects  $\underline{\bar{F}}_i$  at the object coordinates. Define by  $\underline{z}_i$  the coordinates of the frame obtained by translating the end effector frame of arm  $i$  by  $\underline{s}_i$ , i.e. the frame associated with  $\underline{\bar{F}}_i$ . Then by the duality of force/velocity relationships we have

$$\dot{\underline{z}}_i = W_i^T \dot{\underline{z}}_i \quad (3.1)$$

with  $\underline{z}_i$  the end effector coordinates of arm  $i$ , and  $W_i$  is constant, nonsingular and given by (2.8).

Next we solve (2.7) for  $\underline{\bar{F}}$  in terms of  $\underline{P}$  and coordinates  $\underline{P}_r$ , where  $\underline{P}_r$  parameterize the internal object forces, by

$$\underline{\bar{F}} = U \begin{bmatrix} \underline{P} \\ \underline{P}_r \end{bmatrix} = \begin{bmatrix} \overline{W}^+ : V \end{bmatrix} \begin{bmatrix} \underline{P} \\ \underline{P}_r \end{bmatrix} \quad (3.2)$$

with  $U \in \mathbb{R}^{6L \times 6L}$ ,  $\underline{P}_r \in \mathbb{R}^{6(L-1)}$ ,  $\overline{W}^+ \in \mathbb{R}^{6L \times 6}$ ,  $V \in \mathbb{R}^{6L \times 6(L-1)}$ , and  $\overline{W}^+$  is given by  $A\overline{W}^T(\overline{W}A\overline{W}^T)^{-1}$ , with  $A \in \mathbb{R}^{6L \times 6L}$  invertible, and

$$V = \begin{bmatrix} I & 0 & 0 & \dots & 0 \\ -I & I & 0 & & \vdots \\ 0 & -I & I & & \vdots \\ 0 & 0 & \ddots & \ddots & 0 \\ \vdots & & 0 & -I & I \\ 0 & & & 0 & -I \end{bmatrix} \quad (3.3)$$

In (3.3),  $I$  represents the  $6 \times 6$  identity matrix. From (3.2) and (3.3), it may be readily seen that  $V$  is a parametrization of the nullspace of  $\bar{W}$  (note  $\bar{W}V = 0$ ).  $\underline{P}_r$  represents the  $(L-1)$  independent squeezing forces between  $(L-1)$  distinct end effector pairs due to the construction of  $V$ , i.e. components of  $\bar{F}$  which do not contribute to motion but squeeze the object. Furthermore,  $U$  in (3.2) is nonsingular and constant if  $A$  is constant. It is a simple exercise to check that, for

$$A = \begin{bmatrix} A_1 & & 0 \\ & \ddots & \\ 0 & & A_L \end{bmatrix} \quad \text{with} \quad A_i = \begin{bmatrix} \alpha_1^i & & 0 \\ & \ddots & \\ 0 & & \alpha_6^i \end{bmatrix} \quad (3.4)$$

with  $\alpha_j^i$  positive and  $\sum_{j=1}^6 \alpha_j^i = 1$ , then in (3.2) we obtain, with  $\underline{P}_r = 0$ ,  $(\bar{F}_i)_j = \alpha_j^i (\underline{P})_j$ , i.e. we may do load sharing in the ‘pulling together’ sense of (2.10) as before when  $\underline{P}_r = 0$ . This extends the work in [18], where  $A = I$  and  $V$  was simpler for the two arm case.

Again using the force/velocity duality and defining  $\underline{z} = [\underline{z}_1^T, \dots, \underline{z}_L^T]^T$ , we have from (3.2)

$$\begin{bmatrix} \dot{\underline{\phi}} \\ \Delta \dot{\underline{z}} \end{bmatrix} = U^T \dot{\underline{z}} \quad (3.5)$$

where  $\Delta \dot{\underline{z}}$  is the difference in velocities between the frames represented by the  $\underline{z}_i$ ’s, i.e. the velocities associated with  $\underline{P}_r$ .

We may apply the ‘computed torque’ methodology to the system in coordinates  $[\dot{\underline{\phi}}^T \Delta \dot{\underline{z}}^T]^T$  in (3.5). Specifically, referring to (2.9), (3.1) and (3.5), if the second order kinematics of the arms are given by

$$\ddot{\underline{z}} = J_\beta(\underline{\beta}) \ddot{\underline{\beta}} + \dot{\underline{\beta}}^T [H_\beta(\underline{\beta})] \dot{\underline{\beta}} \quad (3.6)$$

then the input of

$$\underline{\tau}_\beta = [M][J]^{-1}[\tilde{W}]^T[U^T]^{-1}\underline{w} + \dot{\underline{\beta}}^T[N]\dot{\underline{\beta}} - J^T \underline{F} - [M][J]^{-1}\dot{\underline{\beta}}^T[H]\dot{\underline{\beta}} \quad (3.7)$$

sets the dynamics (assuming perfect modeling, measurements, etc.) to

$$\begin{bmatrix} \ddot{\underline{\phi}} \\ \Delta \ddot{\underline{z}} \end{bmatrix} = \underline{w} \quad (3.8)$$

(in (3.7),  $\tilde{W} = \begin{bmatrix} W_1 & & 0 \\ & \ddots & \\ 0 & & W_L \end{bmatrix}$ , and it is assumed that  $U$  is constant, although this is not necessary).

Notice that in (3.7) the dependence of  $M$ ,  $J$ ,  $N$  and  $H$  on  $\underline{\beta}$  has been omitted.

Any one of a number of possible strategies may be used to set  $\underline{w}$  (i.e. PD, PID, force feedback). Issues associated with this type of control are well documented ([24] contains a discussion of such issues for joint space and end effector PD control of cooperating arms). It is believed by the authors that the control structure represented by (3.7) is particularly appealing since the coordinates involved go directly to the ‘heart’ of the cooperating arm problem, namely those associated with the object, its motion and its loading which, as we have seen, may be related directly to the overall system loading. The extension at the object coordinates to include  $\underline{P}_r$  allows this to be introduced. This differs from [9] where a feedforward approach is used to control internal forces after system linearization. It also represents a generalization of the framework of [11], in which object motion is controlled together with non-motion causing end effector force components. In (3.8), the extension of coordinates allows the consideration of a wide class of strategies at the object level, due to the exposure of the object motion/squeezing structure represented by (3.8) (and

in fact  $\underline{w}$  may be chosen in (3.8) to implement a strategy similar to that in [11], but with the internal forces controlled being  $\underline{P}$ , instead of end effector force components). Of course, as in all such approaches, effectiveness will be limited by sensory data and computational power available, and for some cases (e.g. where some parameters are not well known) other methods, such as adaptive control methods, will prove useful. However, it is believed irrespectively that further investigation of the control scheme introduced here ((3.7)) will prove both informative and useful. Results will be presented in a future paper.

#### 4. REDUNDANT ARMS IN COOPERATION

If one or more of the arms in the system is kinematically redundant, we are presented with additional problems due to the increased complexity — for the redundant arm(s), we have an overabundance of kinematically independent inputs relative to the end effector(s) of the arm(s). This means that we have multiple solutions for the inverse kinematics, the investigation of which is a large research area of its own; see, for example, [4], [15], [23]. Additionally, the object/end effector spaces do not specify the system totally. Recall in Section 3 we had  $6L$  convenient variables for the  $6L$  actuators — 6 coordinates of object motion, and  $(6 - 1)L$  independent velocity differences between end effector related frames. In the redundant situation, these coordinates remain but now there are  $6L + R$  actuators where  $R$  is the total degree of redundancy in the arms. To control the system we need to specify  $R$  more (independent) variables, representing the choice of arm configurations to be made.

However, the potential benefits of using redundant arms are huge — the self-motion of the arms may be used to avoid collisions with obstacles (and each other), and also the system as a whole gains more mobility, which may be used to reduce actuator demands and aid in cooperative movement. First results in trajectory planning for multiple cooperating arms are presented in [10] and [16], and a generalized inverse of the Jacobian is used in [8] to reduce torque demands while controlling object motion forces.

Here we develop a controller by extending the coordinate space to specify  $R$  extra variables to be controlled independently using the redundant actuator(s). The idea is to make these introduced variables significant to the multiple manipulator problem. Notice that one advantage is that all results related to end effector, or object spaces, still remain pertinent — the kinematic redundancy is ‘decoupled’ from the object. Therefore the  $R$  extra coordinates should be directly associated with the redundant arm(s) (this is clear when it is remembered that their purpose is to specify the configuration(s) of the arm(s)).

This means that we must specify  $R$  coordinates  $\underline{r}$  such that  $\underline{r}$  is independent of  $\underline{\phi}$  and  $\Delta \underline{z}$  (equivalently independent of the end effector coordinates) and that  $\underline{r}$  and  $\underline{\phi}$  (equivalently  $\underline{r}$  and  $\underline{x}$ ) uniquely define the configurations of all arms. Mathematically, we must have  $R$  constraints such that

$$\dot{\underline{r}} = C \dot{\underline{\beta}} \quad (4.1)$$

where  $C \in \mathbb{R}^{R \times (6L+R)}$ , and  $C$  is such that  $B \in \mathbb{R}^{(6L+R) \times (6L+R)}$ , defined by

$$B = \begin{bmatrix} [J] \\ [C] \end{bmatrix} \quad (4.2)$$

is invertible. This will be true provided  $\underline{r}$  is chosen correctly and  $J$  is of full rank (i.e. no arm is in a singular configuration).

Given a choice of  $\underline{r}$ , and noting (4.2), an extension of the control in Section 3 leads to

$$\underline{\tau}_\beta = [M] B^{-1} W^* [\underline{U}^T]^{-1} \underline{w} + \dot{\underline{\beta}}^T [N] \dot{\underline{\beta}} - J^T \underline{F} - [M] B^{-1} \dot{\underline{\beta}}^T [\underline{H}] \dot{\underline{\beta}} \quad (4.3)$$

where  $W^* = \begin{bmatrix} \dot{W} & 0 \\ 0 & I_R \end{bmatrix}$ ,  $\underline{U}^T = \begin{bmatrix} \underline{U}^T & 0 \\ 0 & I_R \end{bmatrix}$ ,  $\dot{\underline{\beta}}^T [\underline{H}] \dot{\underline{\beta}} = \begin{bmatrix} j \\ c \end{bmatrix} \dot{\underline{\beta}}$ , and  $I_R$  is the  $R \times R$  identity matrix) which sets the dynamics to

$$\begin{bmatrix} \ddot{\underline{\phi}} \\ \ddot{\Delta \underline{z}} \\ \ddot{\underline{r}} \end{bmatrix} = \underline{w} \quad (4.4)$$

The comments made in Section 3 apply again to this situation, and if the kinematics have been planned, giving the  $\underline{r}$  trajectories, various strategies may be used to select  $\underline{u}$  to track the trajectory as before. This strategy of extending the workspace coordinates to control redundant manipulators was used by Egeland [4] for single redundant manipulators, and (4.3)-(4.4) represents the extension of this to the multiple manipulator situation.

One further intriguing possibility is that of selecting  $\underline{r}$  trajectories 'on-line' to allow the redundant arms to configure themselves along the trajectory to react to problems unforeseen in the planning, while continuing to track the planned desired object motion and squeezing trajectories. Research in this direction is underway currently.

## 5. CONCLUSIONS

We have presented a framework for the control of multi-arm robot systems. Cooperation is modeled in terms of coordinates representing object motion and internal object squeezing. A control structure has been developed to track trajectories in these coordinates. This was extended to allow the use of redundant arms by making an appropriate extension of coordinates.

### References

- [1] T.E. Alberts and D.I. Soloway, "Force Control of a Multi-Arm Robot System," *Proc. IEEE Conference on Robotics and Automation*, 1988, 1490-1496.
- [2] S. Arimoto, F. Miyazaki, and S. Kawamura, "Cooperative Motion Control of Multiple Robot Arms or Fingers," *Proc. IEEE Conference on Robotics and Automation*, 1987, 1407-1412.
- [3] A. Cole, J. Hauser, and S. Sastry, "Kinematics and Control of Multifingered Hands with Rolling Contact," *Proc. IEEE Conference on Robotics and Automation*, 1988, 228-233.
- [4] O. Egeland, "Task-Space Tracking with Redundant Manipulators," *IEEE Journal of Robotics and Automation*, RA-3, 1987, 471-475.
- [5] R.A. Freeman, "Dynamic Modeling of Robotic Linkage Systems in Terms of Arbitrary Generalized Coordinates," *Proc. IEEE Conference on Systems, Man and Cybernetics*, 1987, 787-797.
- [6] R.A. Freeman and D. Tesar, "Dynamic Modeling for the Overconstrained Mode of Coordinated, Multiple Manipulator Operation," *Proc. Tenth U.S. National Congress of Applied Mechanics*, Austin, TX, June 16-20, 1986.
- [7] S. Hayati, "Hybrid Position/Force Control of Multi-Arm Cooperating Robots," *Proc. IEEE Conference on Robotics and Automation*, 1986, 82-89.
- [8] O. Khatib, "Augmented Object and Reduced Effective Inertia in Robotic Systems," *Proc. 1988 American Control Conference*, 1988, 2140-2147.
- [9] K. Kreutz and A. Lokshin, "Load Balancing and Closed Chain Multiple Arm Control," *Proc. 1988 American Control Conference*, 1988, 2148-2155.
- [10] S. Lee and J.M. Lee, "Task-Oriented Dual-Arm Manipulability and Its Application to Configuration Optimization," *Proc. IEEE Conference on Decision and Control*, Austin, TX, 1988, 2253-2260.
- [11] Z. Li, P. Hsu and S. Sastry, "Dynamic Coordination of a Multiple Robotic System with Point Contact," *Proc. 1988 American Control Conference*, Atlanta, GA, 1988, 505-510.
- [12] M.D. Montemerlo and A. DeYoung, "The Space Perspective: Man-Machine Redundancy in Remote Manipulator Systems," Keynote Speech presented at *NATO Advanced Research Workshop on Robots with Redundancy: Design, Sensing and Control*, Salo, Lago di Garda, Italy, July 1988.
- [13] Y. Nakamura, K. Nagai, and T. Yoshikawa, "Mechanics of Coordinative Manipulation by Multiple Robotic Mechanisms," *Proc. IEEE Conference on Robotics and Automation*, 1987, 991-998.
- [14] M.E. Pittelkau, "Adaptive Load-Sharing Force Control for Two-Arm Manipulators," *Proc. IEEE Conference on Robotics and Automation*, 1988, 498-503.

- [15] I.H. Suh and K.G. Shin, "Coordination of Dual Robot Arms Using Kinematic Redundancy," *Proc. IEEE Conference on Robotics and Automation*, 1988, 504-509.
- [16] T.J. Tarn, A.K. Bejczy, and Z.F. Li, "Dynamic Workspace Analysis of Two Cooperating Robot Arms," *Proc. 1988 American Control Conference*, Atlanta, GA 1988, 489-498.
- [17] T.J. Tarn, A.K. Bejczy, and X. Yun, "Design of Dynamic Control of Two Cooperating Robot Arms: Closed Chain Formulation," *Proc. IEEE Conference on Robotics and Automation*, 1987, 1-7.
- [18] M. Uchiyama and P. Dauchez, "A Symmetric Hybrid Position/Force Control Scheme for the Coordination of Two Robots," *Proc. IEEE Conference on Robotics and Automation*, 1988, 350-356.
- [19] M.A. Unseren and A.J. Koivo, "Kinematic Relations and Dynamic Modeling for Two Cooperating Manipulators in Assembly," *Proc. 1987 IEEE Conference on Systems, Man and Cybernetics*, Washington, DC, 1987, 798-802.
- [20] I.D. Walker, R.A. Freeman, and S.I. Marcus, "Dynamic Task Distribution for Multiple Cooperating Robot Manipulators," *Proc. IEEE Conference on Robotics and Automation*, 1988, 1288-1290.
- [21] I.D. Walker, R.A. Freeman, and S.I. Marcus, "Distribution of Dynamic Loads for Multiple Cooperating Robot Manipulators," *Journal of Robotic Systems*, February 1989.
- [22] I.D. Walker, R.A. Freeman, and S.I. Marcus, "Internal Object Loading for Multiple Cooperating Robot Manipulators," submitted to *IEEE Conference on Robotics and Automation*, 1989.
- [23] I.D. Walker and S.I. Marcus, "Subtask Performance by Redundancy Resolution for Redundant Robot Manipulators," *IEEE Journal of Robotics and Automation*, 4, June 1988, 350-354.
- [24] J.T. Wen and K. Kreutz, "Stability Analysis of Multiple Rigid Robot Manipulators Holding a Common Rigid Object," *Proc. IEEE Conference on Decision and Control*, Austin, TX, 1988, 192-197.
- [25] T. Yoshikawa and K. Nagai, "Manipulating and Grasping Forces in Manipulation by Multifingered Hands," *Proc. IEEE Conference on Robotics and Automation*, 1987, 1998-2004.
- [26] Y.F. Zheng and J.Y.S. Luh, "Optimal Load Distribution for Two Industrial Robots Handling a Single Object," *Proc. IEEE Conference on Robotics and Automation*, 1988, 344-349.
- [27] NASA Technical Memorandum 100989, "Advancing Automation and Robotics Technology for the Space Station and for the U.S. Economy," Progress Report, October 1987 - March 1988.

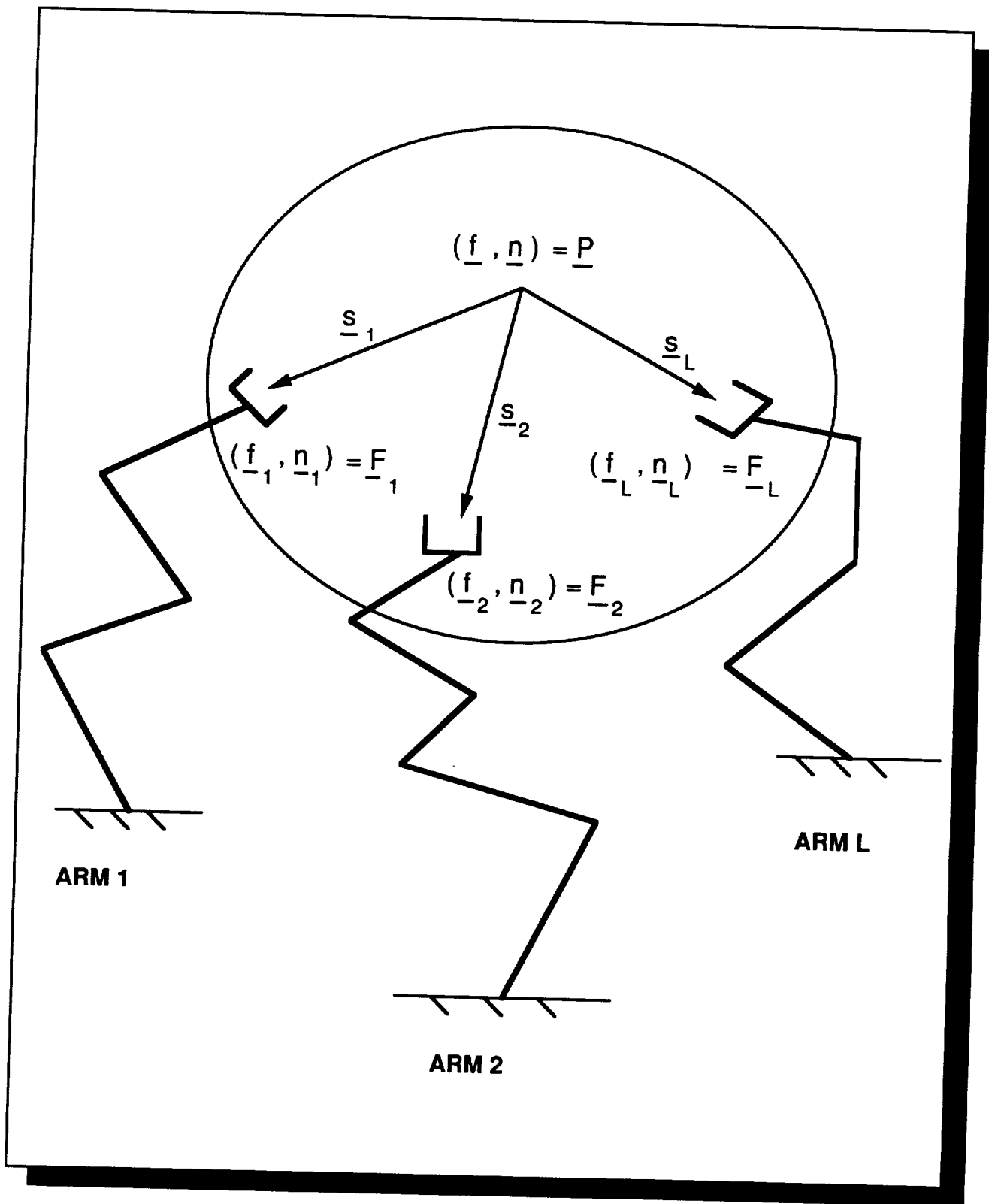


Figure 1. Multiple arm cooperating system.

## Reflexive Obstacle Avoidance for Kinematically-Redundant Manipulators

James P. Karlen, Jack M. Thompson, Jr., James D. Farrell, Håvard I. Vold  
Robotics Research Corporation  
5400 DuPont Circle, TechneCenter  
Milford, Ohio 45150

### Abstract

*Dexterous telerobots incorporating 17 or more degrees of freedom operating under coordinated, sensor-driven computer control will play important roles in future space operations. They will also be used on Earth in assignments like fire fighting, construction and battlefield support. A real time, reflexive obstacle avoidance system, seen as a functional requirement for such massively redundant manipulators, was developed using arm-mounted proximity sensors to control manipulator pose. The project involved a review and analysis of alternative proximity sensor technologies for space applications, the development of a general-purpose algorithm for synthesizing sensor inputs, and the implementation of a prototypical system for demonstration and testing. A 7 DOF Robotics Research K-2107HR manipulator was outfitted with ultrasonic proximity sensors as a testbed, and Robotics Research's standard redundant motion control algorithm was modified such that an object detected by sensor arrays located at the elbow effectively applies a "force" to the manipulator elbow, normal to the axis. The arm is "repelled" by objects detected by the sensors, causing the robot to steer around objects in the workspace automatically while continuing to move its tool along the commanded path without interruption. The mathematical approach formulated for synthesizing sensor inputs can be employed for redundant robots of any kinematic configuration. The work described in this paper was funded by NASA Langley Research Center.*

### 1.0 INTRODUCTION

The National Aeronautics and Space Administration (NASA) has identified a number of promising applications for advanced robots and telerobots in future space operations. The most sophisticated of these robots will be required to perform complex tool-handling tasks with dexterity approaching "man-equivalence", while operating with a minimum of human intervention.

One class of NASA telerobots will be designed for EVA operations in high vacuum, zero-G or micro-G environments. Prototypical of this class is the Flight Telerobotic Servicer (FTS), a general-purpose tool-handling robot that will be used by Astronauts to assist in the assembly and servicing of the U.S. Space Station. (An illustration of the Grumman-Robotics Research-TRW team concept for FTS is shown in Figure 1.) Transported by NASA's Orbital Maneuvering Vehicle (OMV), the FTS and derivative models will eventually be deployed for remote servicing of the polar orbiting platform and the growing fleet of civil and military satellites in geosynchronous orbit. EVA servicing robots like the FTS also may become standard integral maintenance and repair subsystems on board large unmanned space probes and manned interplanetary vessels.

Robots with capabilities similar to the Flight Telerobotic Servicer will play important roles in initial exploration of the surface of other planets and moons in the Solar system. Designed to operate in the local gravity field and atmosphere, these units will naturally be built with somewhat different physical proportions, materials and sensor systems than high-vacuum, zero-G telerobots like FTS. They will also,

by necessity, be capable of considerable local intelligence and autonomy, since transmission delays preclude real-time control from remote human operators. In the planned Mars Rover mission, for instance, mobile robots will be used to survey Mars autonomously, examining the surface at close range and retrieving geological samples for analysis on Earth. Other devices of this type will be used to resume exploration of the surface of the Moon, and to survey the satellites of Jupiter and Saturn.

A third class of dexterous tool-handling robots are being planned by NASA for use in micro-G, IVA operations. These devices will operate inside Space Station laboratory modules, initially performing routine material-handling functions such as those required in semiconductor processing and liquid pharmaceutical processing operations which exploit the micro-G environment. Very general-purpose mobile servicers may ultimately be developed to handle "housekeeping" chores within the crew modules of the Space Station and manned interplanetary vessels.



Figure 1:  
*Grumman-TRW-Robotics Research  
Concept for NASA Flight Telerobotic Servicer*

The specific physical configurations of NASA's dexterous telerobots will differ from one application to another, and one working environment to another. Some may have a single tool-handling arm. Some may have two dexterous arms mounted on a torso/waist assembly. Others may utilize "spider" configurations to provide mobility on a truss, including several arms, each with a dexterous end-effector, plus a number of legs and peripheral camera and light positioners. The control system capabilities of these robots will also differ from one application to another. Some will function primarily in a teleoperated or shared control mode, with the "man in the loop", while others will operate autonomously for long periods of time. Nevertheless, the basic missions established for NASA's dexterous manipulator systems impose certain common requirements for their physical designs and control architectures. One can reasonably foresee the following:

**The Need for Kinematic Redundancy** In order to perform their assigned tool-handling tasks, NASA's dexterous telerobots will generally have more than six joints operating under simultaneous, coordinated control. Seven axes are the minimum required in a mechanical manipulator to emulate the basic motions of the human arm, from the shoulder to the wrist. Like the human arm, a 7 degree of freedom (DOF) manipulator can assume any number of different joint configurations, or arm poses, for a given position and orientation of the "hand" (or "toolpoint"). With one "redundant" degree of freedom, the manipulator arm has the freedom, for example, to reach around and avoid collisions with objects in certain locations in its workspace while it performs its programmed task. In principle, the more redundant degrees of freedom incorporated in the manipulator system, the more versatility it has. To perform extremely complex tasks with skill approaching "man equivalence", future space telerobots will eventually incorporate hundreds of degrees of freedom operating under coordinated control.

**The Need for Sensor-Driven Computer Control of Redundancy** A computer will be required to coordinate the actions of the joints in these redundant manipulator systems. In the limited number of mission scenarios in which such telerobots might be commanded in a teleoperated mode by nearby Astronauts, it is unlikely that many of the manipulator configurations could be slaved to a "replica" master (a device whose geometry matches that of the slave which permits the human operator to directly and continuously control the individual joints in the slave manipulator). Obviously, in massively redundant manipulator systems and ones which assume non-anthropomorphic configurations, this would be entirely impractical. Instead,



one can reasonably predict that the human operator will share the control of the dexterous manipulator with a computer, where, at most, the human essentially "flies the hands" of the manipulator system by specifying a 6 DOF goalpoint, while the local computer decides how to move the manipulator joints to execute this toolpoint trajectory. (In this mode, the human operator will also need the ability to control the position of the hand in some tool axes and the forces applied in the others.)

The local computer system which is charged with real-time, coordinated control of manipulator joints will decide how to employ the redundancy in the system based on sensory inputs about its internal condition and its environment, using a set of rules which seek to optimize the situation. The NASREM architecture, developed by Dr. James S. Albus, et al, at the U.S. National Institute of Standards and Technology (formerly National Bureau of Standards), has been adopted by NASA as its standard reference model for advanced telerobot control systems<sup>32</sup>. NASREM utilizes a hierarchical architecture in which each successively higher level has a broader purview with respect to space and time, and is equipped with sensory feedback, memory and logical functions appropriate to its level of responsibility. In this context, authority over how to use the kinematic redundancy in the manipulator will not reside within any single level of the control system, but will be affected by decisions made at all levels.

Robotics Research Corporation is principally concerned with those levels of the telerobot control system responsible for making "reflexive motion control" decisions based on local, kinesthetic sensors mounted on the manipulator. These might be viewed as "brainstem" functions, analogous to the autonomic or sympathetic divisions of the central nervous system in biological models. (They are encompassed by Levels 1, 2 and 3 in the NASREM model-- "Servo", "Primitive", and "Elemental Move".)

Investigators at Robotics Research believe that this reflexive motion control system will employ a hierarchy of competing rules, or objective functions, in making a balanced decision each clock cycle about how best to dispose manipulator redundancy. We propose that, in general, the robot should attempt to execute the commanded toolpoint trajectory,

1. while avoiding collisions with itself, and
2. while avoiding collisions with objects that are detected in the telerobot's working envelope, and
3. while recognizing singularities intrinsic to its mechanical geometry and using them appropriately,
  - a) to produce energy-efficient, graceful motion, or
  - b) to increase leverage (mechanical advantage), or
  - c) to control "impedance" at the toolpoint, and
4. while "favoring" any joints that are sensed to be closer to their thermal limits than others.

Obviously, a higher level in the hierarchical control system may elect to override or reprioritize these objectives based on its broader view of the situation.

Robotics Research Corporation introduced its first products in 1984-- the K-Series line of Dexterous Manipulators and the Type 1 Motion Controller, a motion control system based on the National Bureau of Standards hierarchical architecture and designed specifically to provide real time control of the company's kinematically-redundant arms. The 16-bit Type 1 Motion Controller, and the newer, 32-bit Type 2 model, employ proprietary algorithms which coordinate the joint motions of a redundant system using a set of weighted objective functions and which solve these equations in an extremely computationally-efficient manner<sup>30,31</sup>. Original algorithms accomplish 3a, above, i.e., they recognize singularities intrinsic to the manipulator's mechanical geometry and use them appropriately to produce efficient, graceful motion. Robotics Research has subsequently been developing new objective functions compatible with these algorithms.

The goal of the research described in this report was to implement objective function 2, above-- a means for real-time control of manipulator redundancy using arm-mounted proximity sensors to provide reflexive collision avoidance. While this effort aimed primarily at avoiding collisions with external objects detected in the workspace, in fact, it is clear that the principle devised applies equally well to function 1, above, i.e., detecting and avoiding collisions between different members of the manipulator itself.

## **2.0 TECHNICAL OBJECTIVES**

The research effort described in this report had four specific objectives:

1. To survey alternative proximity sensor technologies that could be used on dexterous manipulators to accomplish real-time, reflexive obstacle avoidance, with particular emphasis on sensor systems that could be employed in the space environment;
2. To develop algorithms which translate arm-mounted proximity sensor data into appropriate penalty functions representing obstacles in the robot workspace;
3. To modify Robotics Research Corporation's existing software to synthesize a set of motion commands which automatically cause a kinematically-redundant manipulator to avoid obstacles while accomplishing a prescribed end-effector path;
4. To implement such a reflexive obstacle avoidance system which controls the redundant degree of freedom of an available Robotics Research K-2107HR 7-axis manipulator (Figure 2) and to demonstrate the ability of this system to execute prescribed toolpoint paths while automatically keeping the elbow clear of obstacles placed within its workplace.

## **3.0 SURVEY OF APPLICABLE PROXIMITY SENSOR TECHNOLOGY**

### **3.1 Functional Requirements for Arm-Mounted Proximity Sensors**

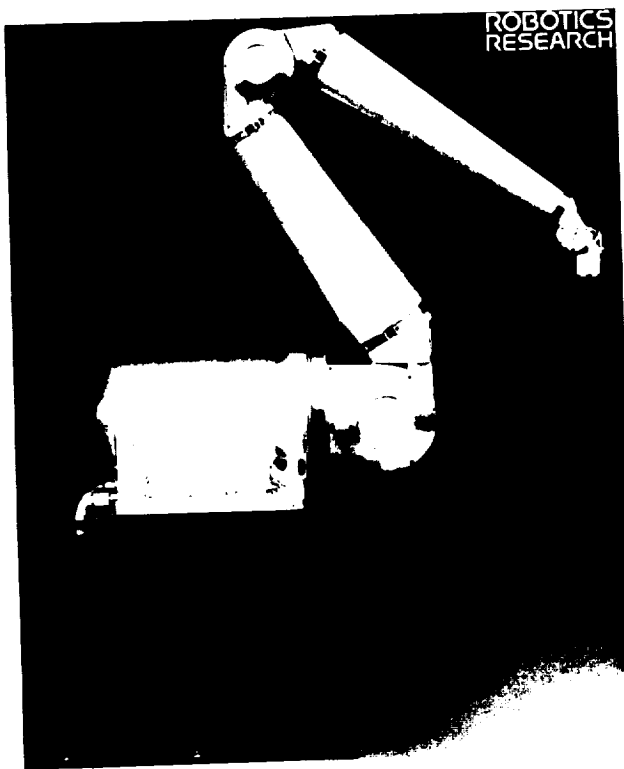
For the general problem of reflexive obstacle avoidance using arm-mounted sensors, a system is needed that effectively creates a "field" around the entire manipulator assembly capable of detecting the presence and measuring the coordinates of objects anywhere close to the surface of the unit (Figure 3).

We established the following specific guidelines in evaluating alternative sensor technologies:

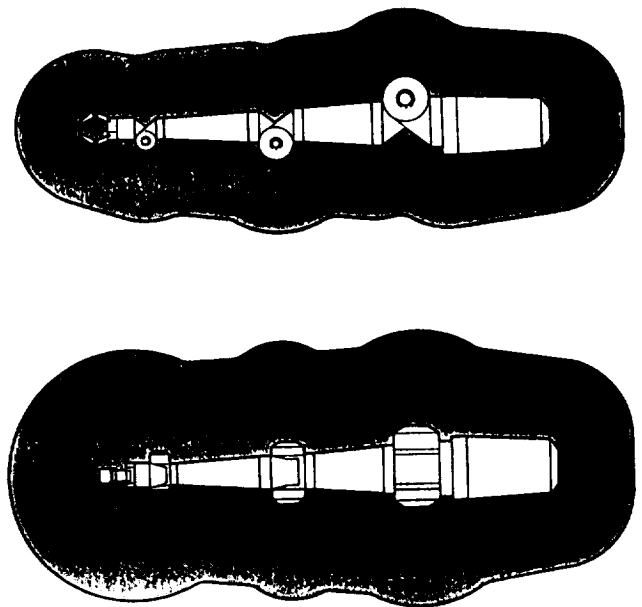
1. The system must have the ability to detect objects of a wide variety of physical sizes, geometries, materials, surface finishes and temperatures;
2. Candidate sensor hardware must permit compact mounting on the manipulator in array configurations which, given the intrinsic beam geometry, provide full coverage (i.e., no blind spots);
3. Candidate sensor hardware must be capable of reliably detecting and measuring the distance of objects normal to the manipulator surface at a minimum range of one inch to 12 inches with a minimum accuracy of  $\pm 10\%$  (a zero to 24 inch range is considered to be ideal for the obstacle avoidance application);
4. The proximity sensor array covering the entire manipulator assembly should operate with an update rate of less than 20 milliseconds (50 Hz), including transmission delays and computation;
5. Sensor-emitted energy must not cause injury to personnel or damage to equipment within or without its effective sensing range;
6. The system should not have moving mechanical parts (e.g., no scanning mechanism should be used to generate a useful field of view).

Arrays of ultrasonic acoustic sensors might be devised which meet these qualifications for Earth use, as discussed in a subsequent section of this report. In space, candidate sensors obviously must employ some frequency in the electromagnetic spectrum, but a number of other requirements must also be considered:

1. The sensor hardware must have the ability,
  - a) to withstand the space environment (i.e., to tolerate high vacuum, ambient radiation, thermal extremes, and shock and vibration), and
  - b) to function properly in that environment (i.e., not to be confused by solar radiation and EMI from other sources);
2. Sensor-emitted radiation must not interfere with other spacecraft systems;
3. The sensor system must consume little power;
4. Any "blanket" of proximity sensors must not adversely affect the manipulator's ability to reject waste heat to space.



*Figure 2:  
Robotics Research K-2107HR  
7 DOF Manipulator Arm*



*Figure 3:  
Idealized Proximity-Sensing "Field"  
Surrounding Manipulator*

### 3.2 Proximity Sensor Principles of Operation

In order to drive the reflexive obstacle avoidance algorithms, the sensor system must provide information that describes the location of the obstacle relative to the manipulator. Location can be reduced to components of bearing, azimuth and range. Means of deducing the bearing and azimuth are, as follows:

1. A dense blanket of simple transmitter/receivers, radially mounted, each dedicated to sensing a sector of bearing and azimuth (Figure 4);
2. A system of numerous emitters and one-dimensional sensor arrays, radially or laterally mounted, each

equipped with an optical system which couples each sensor pixel to a particular sector of space (Figure 5);

3. A system of a few emitters and two-dimensional sensor arrays, radially or laterally mounted, each equipped with an optical system which couples every sensor pixel to a particular sector of space (Figures 6 and 7).

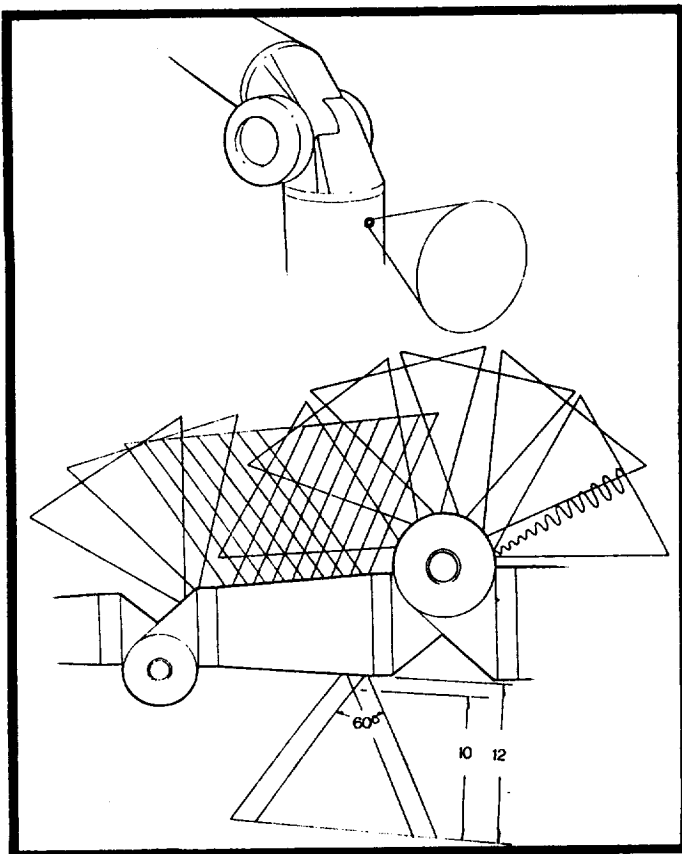
Independent of the means employed to determine bearing and azimuth, the sensors and emitters of illumination can utilize a variety of techniques to deduce range.

1. Echo Intensity Pulsed or continuous radiation is transmitted and the intensity of the return echo reflected from an object is measured. Since the intensity of the reflection for a point source diminishes at the inverse square of the distance, a range can be calculated. Due to variations in reflectivity of different targets (size, geometry, surface finish), echo intensity is not a reliable general-purpose ranging system.
2. Time of Flight The time delay between the transmission of a signal pulse and the return echo reflected from an object is measured. Distance is directly proportional to time delay.
3. Amplitude Modulation Continuous amplitude-modulated radiation is transmitted (its intensity is varied cyclically) and the phase shift of the return echo is compared with that of the emitted reference signal. Phase shift is a function of distance. The significant advantages of the phase shift approach are that the electronics required for continuous emission are relatively simple compared to pulse-type systems and it is a superior technique for use in measuring distance at very short ranges. A factor that must be taken into account with phase shift systems is the ambiguity which arises when signals are returned from a distance that exceeds one-half the speed of light divided by the modulation frequency. The intensity may be employed to resolve this problem by the use of an appropriate intensity threshold (below which the signal is disregarded) and the choice of a reasonably long ambiguity distance.
4. Frequency Modulation Continuous frequency-modulated radiation is transmitted (its frequency is varied cyclically) and the phase shift of the return echo is compared with that of the emitted reference signal. Phase shift is a function of distance. Ambiguity is again a factor.
5. Triangulation Bearing and azimuth information to the same object from different points of known location and separation can be used to determine range. If the angular information is discretized into sectors, the range information must be discretized as well.

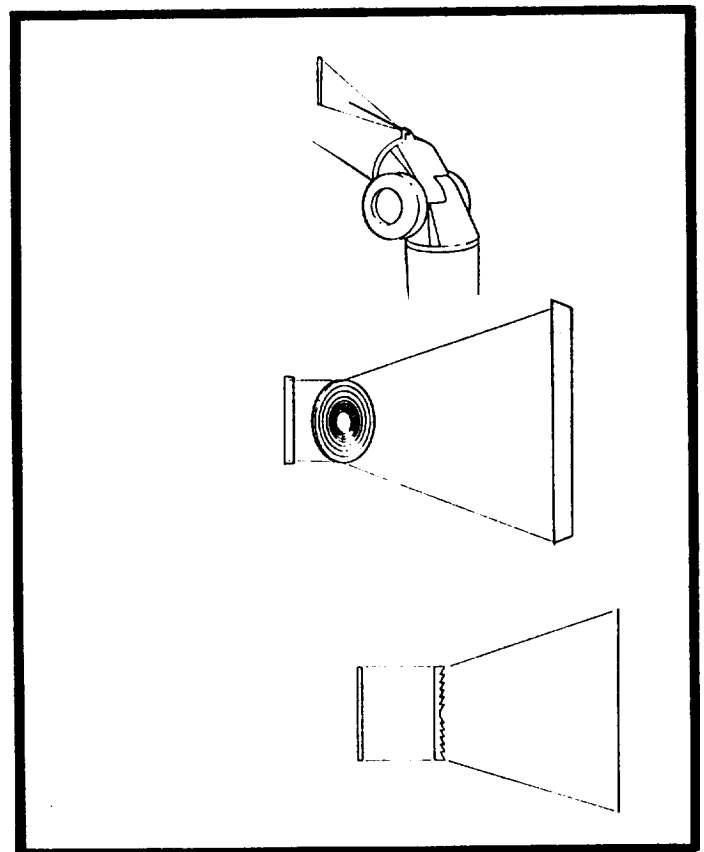
### 3.3 Candidate Sensors for Space Applications

A number of available sensor technologies might serve the purpose in a reflexive obstacle avoidance system for space applications. The most promising band of wavelengths for the system ranges from the near-infrared (1100 nm) to the near-ultraviolet (200 nm). Silicon-based photosensitive devices, in particular, are suitable for this part of the spectrum. Gallium arsenide-based devices also could be employed and would be desirable if the telerobot were subjected to an intense radiation environment. However, gallium arsenide-based devices are expensive and offer fewer options for large scale integration with local signal conditioning, logic and multiplexing electronics.

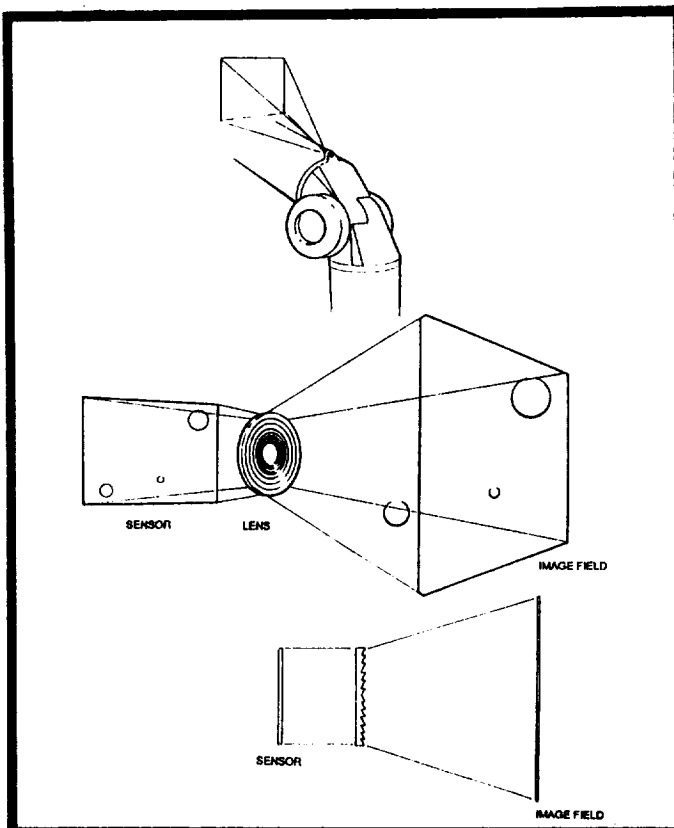
This sensor system must be able to function reliably independent of the background radiation. Intense solar illumination, Earthlight and moonlight, reflections and emissions from nearby spacecraft, and the extreme contrast against the blackness of space make this a challenging problem. Photosensitive devices will almost certainly have to extract a usable return signal from background noise by employing controlled illumination. Two techniques, in the opinion of the investigators, offer the most potential: notch pass filtering of the detector, and amplitude modulation of the emitted lighting. Well-known techniques exist for



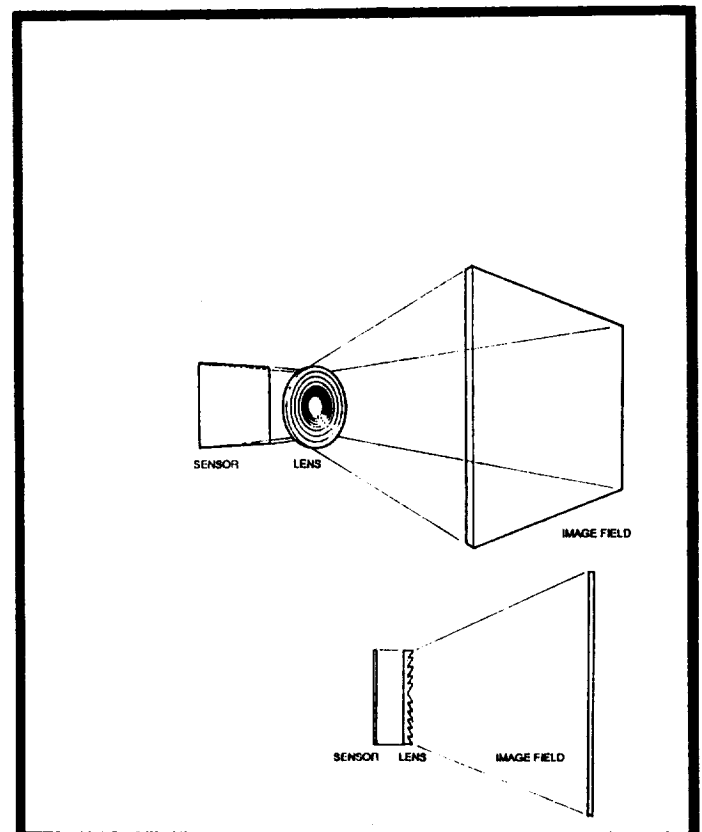
**Figure 4:**  
*Blanket of Transmitter/Receivers  
Mounted Radially*



**Figure 5:**  
*One-dimensional Sensor Array*



**Figure 6:**  
*Two-dimensional Sensor Array*



**Figure 7:**  
*Fresnel Lens Implementation*

filtering the light incident on the detector to allow only a relatively narrow slice of the spectrum, centered on the wavelength of the emitted light, to fall on the detector. This is helpful in limiting the DC response and saturation of the detector when exposed to intense background noise. Likely devices for generating the light for illumination of the objects have very narrow emission spectra and can provide usable levels of illumination relative to the energy content of the incident light at that wavelength.

A second enabling technique for enhancing the general signal-to-noise ratio is amplitude-modulation of the emitted illumination, also a convenient method for determining range. (The frequency corresponding to a fifteen foot ambiguity of range is about  $3.4 \times 10^7$  Hz.) The receiving electronics can be AC-coupled and responsive only to detected signals varying in amplitude at that frequency, providing complete rejection of ambient light signals. Range deduction is accomplished by phase comparison with the emitted illumination.

This general implementation of an arm-mounted proximity sensor system promises to satisfy the key functional requirements for a reflexive obstacle avoidance in space applications. Utilization of the amplitude-modulation technique definitely favors a sensor with fast response and low hysteresis. The significant changes in background lighting, even with filtering, favors a sensor with linearity over a wide dynamic range and relatively low internal gain. Since relatively low voltages are advantageous (consistent with highly integrated, semiconductor electronics), the sensor of choice becomes the silicon photodiode.

To obtain the best linearity and dynamic range, photovoltaic operation is preferred. Photovoltaic operation also minimizes the noise perceived from several sources. Given the need for high speed operation with an amplitude-modulated light source, the proposed implementation of the photodiode is in a photoconductive mode with a reverse bias applied to the photodiode. This system, with a transimpedance amplifier, provides high speed operation with a wide dynamic range. The reverse bias causes a significant reduction in the junction capacitance, the major impediment to high frequency response.

Silicon photodiodes are available in all of the physical arrangements previously discussed (single element sensors of various sizes, linear arrays of discrete photodiodes and rectangular discrete arrays). In addition, silicon photodiode technology can be used to make position sensors. This implementation can be used to determine both analog intensity of perceived light and analog position of the center of the spot of light on the sensor surface, with both one dimensional position on a linear element or both x and y positions on a square or rectangular element. These devices are most commonly employed in laser triangulation probes (refer to Figures 6 and 7). The relatively coarse measurement resolution required in this application favors the use of discrete element arrays, since they provide sufficient resolution (discrete photodiode elements are available as small as 0.004" on a side) and can be fabricated in a IC fashion with amplification, local logical processing and multiplexing components on a single chip.

Such a chip could be designed to report over any suitable network a message that an object has been detected in a particular direction (the chip/element address, corresponding to a particular sector in bearing and azimuth relative to a known point on the manipulator) and at a particular range, deduced locally by comparing the perceived amplitude modulation phase with the emitted light reference signal. This approach would permit the obstacle sensing systems to be placed on a flexible circuit board with a minimum number of electrical connections for power, communications and ground/common. (A higher speed communication system might also be implemented with sync lines or a parallel structure.) In this configuration, all of the high speed operations are resident on the individual chips.

The best candidates for providing amplitude-modulated illumination are solid state laser diodes and light emitting diodes. Both of these devices, closely related, are commonly utilized in fiber-optic communications systems operating at extremely high modulation frequencies. The effective detection of objects and safety of Astronaut vision are both enhanced by using relatively diffuse beams of illumination. Further investigation and design by those skilled in the art will be necessary to choose the intensity of illumination and detailed characteristics of the detectors, amplifiers and other elements of a practical sensor system of this type.

### 3.4 Selection of Proximity Sensors for Experimental Purposes

Several commercial ultrasonic sensor systems were identified which operate over the range required for our laboratory use in algorithms development and demonstrations. After review, sensors and a multiplexing system were selected which operate in a 5"-to-36" range at 225 kHz with an accuracy of  $\pm 0.004$ ". The sensor head measures 1-5/16" long by 5/8" in diameter. The sensor produces an average beam angle over the range of approximately 18 degrees. The multiplexing electronics provide a sequential scanning mode operating at 12 mS per sensor, enabling us to mount a number of sensors close together as an array without interference. Eight multiplexed ultrasonic proximity sensors were mounted in a hemispherical array on either side of the elbow joint, scanning a large "conelike" region around the manipulator elbow (Figures 8 and 9). Each sensor responds to objects in the range of approximately 5" to 26" from the manipulator.

The selected array configuration has gaps in the sensing zone due to the rather narrow field of view of each sensor. An attempt was made to increase the field of view of these sensors by using a concave surface aligned to one side of the beam angle. Although this did increase the beam angle (at the expense of the sensing distance), the approach also had a significant adverse effect on the sensitivity and accuracy of the measurements. In experiments, these gaps ultimately proved to have no significant effect on performance.

## 4.0 CONTROL SYSTEM

### 4.1 Robotics Research Corporation Motion Control Algorithms

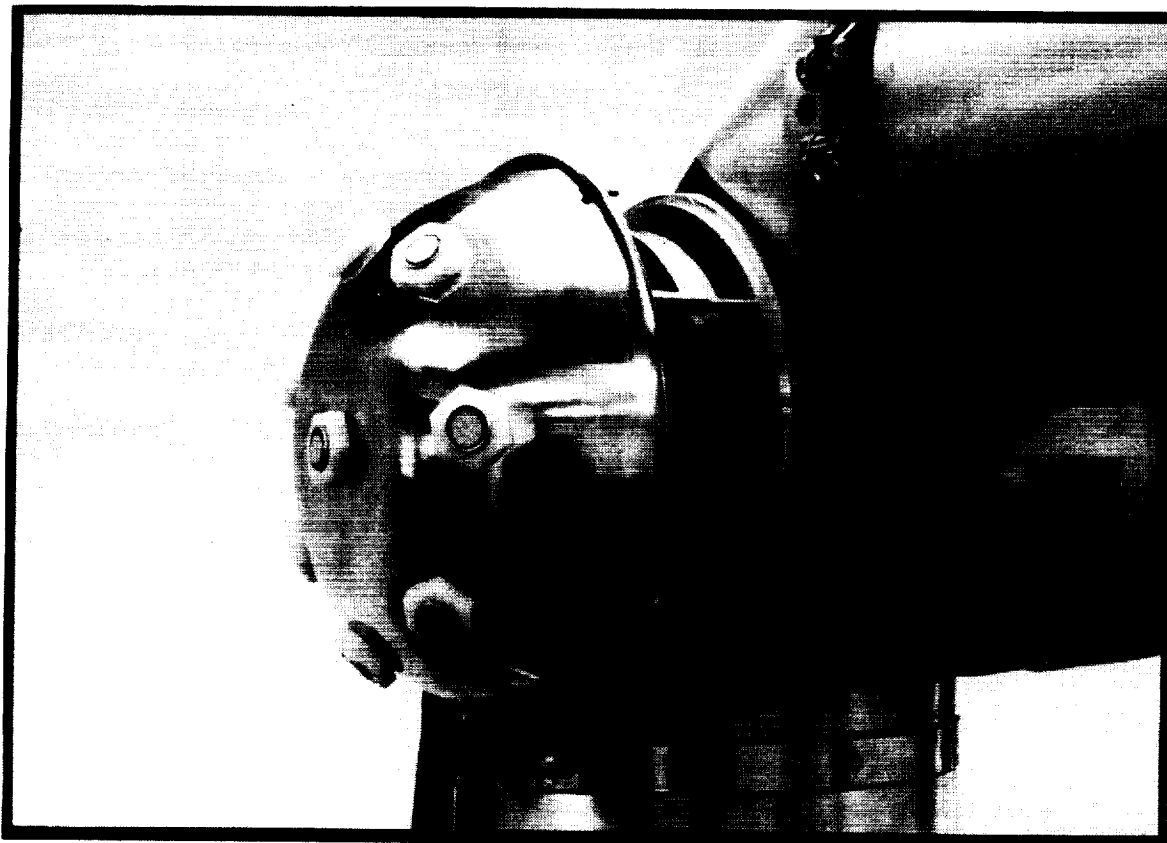
A 7 degree of freedom Robotics Research K-2107HR Dexterous Manipulator and Type 2 Motion Controller were utilized as a testbed for algorithm implementation (Figure 2). The Type 2 Motion Controller is an open-architecture, 32-bit multiprocessor position control system designed to coordinate the motion of a redundant manipulator. A set of weighted objective functions are used to determine the joint motion commands. Algorithms previously implemented by Robotics Research produce graceful, singularity-free motion by treating the redundant system as a spring-loaded mechanism which can deform elastically according to how the end effector is positioned.

Robotics Research's approach to obstacle avoidance employs a similar conceptual model, creating a repellant "force field" in which the manipulator senses obstacles as repellent forces that push on the springs to achieve equilibrium. The intensity of these forces varies with the distance of the object from the arm. The moments generated by these forces cause the system to employ its redundancy to escape the force field.

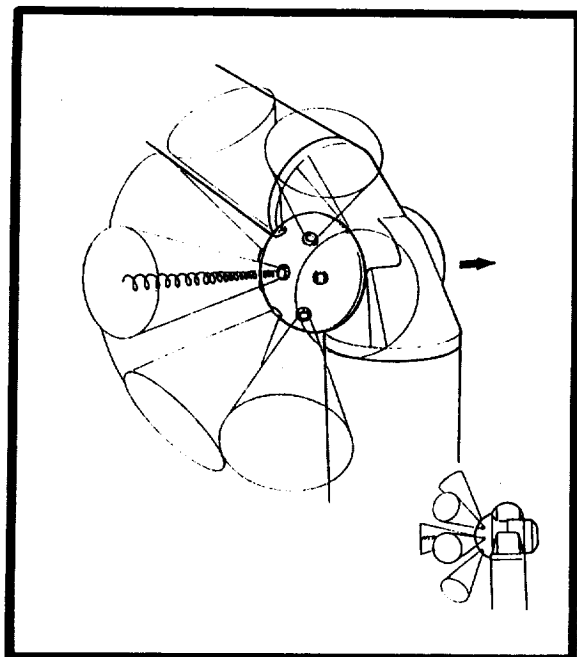
In the case of the K-2107 Dexterous Manipulator used as a testbed for this development program, we have a seven jointed linkage effectively pinned at both ends (fixed at the base and maintaining a commanded position at the tool tip) in which elbow attitude is the only variable (Figure 10). While maintaining a specified tool position and orientation, this "extra" degree of freedom can be used to revolve or "orbit" the elbow in a direction along the centerline of the elbow pitch joint (J4). This capability exists whether the toolpoint remains fixed during the orbit move or is in transit from one goalpoint to another.

It is important to note that a more highly redundant system, such as Robotics Research's new K/B-2017 Dexterous Manipulator, provides for considerably greater freedom of action in avoiding obstacles. The K/B-2017 is seen as prototypical of many future space servicing robots. It incorporates 17 degrees of freedom operating under coordinated computer control, with two 7 DOF arms mounted on a 3 DOF torso/waist. In this system, five independent "orbit" modes can be brought into play simultaneously, one for each elbow and three for the torso. This level of redundancy begins to approach "man-equivalent" versatility and maneuverability when performing complex tool-handling operations in crowded worksites without unintended collisions.

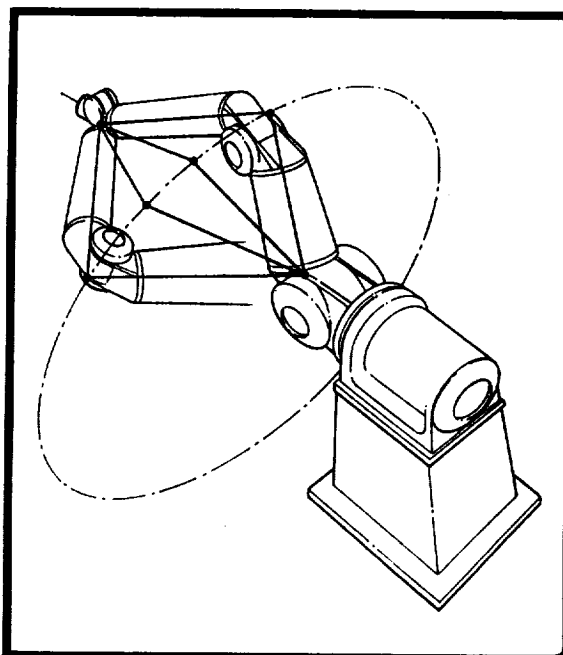
In our laboratory implementation of the proximity sensors system on a 7 degree of freedom arm, signals transmitted by each sensor in the elbow-mounted arrays to the control unit are processed to add vectorially



*Figure 8:  
Elbow-mounted Proximity Sensor Array*



*Figure 9:  
Sensor Array Effective Geometry*



*Figure 10:  
Elbow "Orbit" Move  
with 7 DOF Arm*



the largest forces (as reflected by the closest objects) on either side of the manipulator and to apply the resultant force along the centerline of the elbow (J4). This scheme enables the arm to center itself between objects detected on either side of the elbow joint or on both sides simultaneously. In addition, the forcing function varies exponentially with distance, so that objects detected at close range cause a much more rapid movement of the arm away from an impending collision. This factor can be easily adjusted to increase or decrease the sensitivity of the arm to an object in its working space.

## **5.0 CONCLUSIONS**

The concept of a reflexive, proximity sensor-based, real-time obstacle avoidance system is seen by Robotics Research as one of the key enabling technologies required to exploit fully the intrinsic advantages of redundancy. This NASA-sponsored research project has afforded us the opportunity to demonstrate that concept. The system we have developed, while experimental, works quite well and was implemented without any significant difficulties. Our general conclusions are, as follows:

1. Technology appears to be available today to produce proximity sensor hardware for use in a space environment to support reflexive obstacle avoidance. The practicality of such a system will depend heavily on clever systems integration. It appears to us that a key requirement is the development of a small, rugged, highly integrated emitter-sensor package with local processing.
2. Ultrasonic sensor systems are also available today that could be effective in this application for terrestrial/atmospheric use. Again, the practicality of such a system will depend upon careful systems integration.
3. The placement and coverage of the sensor array used to update the control, in combination with the "spring constant" established for a perceived object, are judged to be the most important design variables affecting the behavior of the obstacle avoidance system.
4. The mathematical approach employed by Robotics Research to translate proximity sensor inputs into additional redundant control criteria is extensible to massively redundant systems of any topology. The more redundant the system in question, the more valuable sensor-driven reflexive obstacle avoidance becomes.
5. We believe this work establishes a theoretical basis for a practical reflexive obstacle avoidance system for future telerobots used both in space and in ground applications. In the case of complex space and nuclear servicing robots, it may, indeed, be impossible to perform the planned operations without some type of reflexive system. Also, real-time collision avoidance is a critical safety subsystem. We further anticipate that opportunities for the application of redundant robots in industrial factory-automation would also be substantially expanded if a reflexive obstacle avoidance system became commercially available. Off-line programming for redundant manipulators and associated workcell design time would be greatly reduced were an obstacle avoidance system to select arm pose dynamically, and without explicit programming or operator intervention.

## **6.0 REFERENCES**

1. Millard, D.L., "Animate Sensing for Industrial Automation Safety", Proceedings from ROBOTS 11, 17th International Symposium on Industrial Robots, pp. 9/23-9/43, April 26-30, 1987, Chicago, IL.
2. Helmers, Carl, "Active Remote Sensing", SENSORS, July 1987, p.89.
3. Graham, James H., Millard, D.L., "Toward Development of Inherently Safe Robots", Proceedings from ROBOTS 11, 17th International Symposium on Industrial Robots, pp. 9/11-9/21, April 26-30, 1987, Chicago, IL.
4. Alpha Industries, Inc., "Theory, Operation, and Application of Microwave Motion Sensing Modules", SENSORS, December 1987, pp.29-36.

5. Everett, H.R., "Noncontact Ranging Systems for Mobile Robots", *SENSORS*, April 1987, pp 9-19.
6. Stefanides, E.J., "Mini Motion-Detector Senses IR With Piezo Film 'Retina'", *Design News*, April 18, 1988, pp 88-91.
7. Biber, C., Ellin, S., Shenk, E., "The Polaroid Ultrasonic Ranging System", *Proceeds 67th Convention of Audio Engineering Society*, October 31, 1980, New York.
8. Irwin, C.T., Caughman, D.O., "Intelligent Robotic Integrated Ultrasonic System", *Proceedings from ROBOTS 9*, Volume 2, pp. 19/38-19/47, June 2-6, 1985, Detroit, Michigan.
9. Jorgensen, C., Hamel, W., Weisbin, C., "Autonomous Robot Navigation", *BYTE*, January 1986, pp. 223-235.
10. Everett, H.R., "A Multielement Ultrasonic Ranging Array", *Robotics Age*, July 1985, pp 13-20.
11. Campbell, Dean, "Ultrasonic Noncontact Dimensional Measurement, *SENSORS*, July 1986, pp. 37-48.
12. Damm, Lennart, "A Three Dimensional Fiber Optical Proximity Sensor System", *Proceedings from ROBOTS 11*, 17th International Symposium on Industrial Robots, pp. 2/1-2/18, April 26-30, 1987, Chicago, IL.
13. Thiele, A.W., Gjellum, D.E., Rattner, R.H., Manouchehri, D., "Sensor Systems Testbed for Telerobotic Navigation", *Proceedings from the Workshop on Space Telerobotics*, JPL Publication 87-13, Volume II, pp. 19-22, July 1, 1987.
14. Amerace Corp., "Ultrasonic Sensor Solves Wire Mesh Transparency Problems", *SENSORS*, July 1987, pp 34-37.
15. Squire, Patty L., "Piezoelectric PVDF Ultrasonic Transducers", *SENSORS*, July 1986, pp 12-16.
16. Johnson, R.F., "A Refresher in Position Sensing", *SENSORS*, September 1987, pp. 18-24.
17. Drumheller, M., "Mobile Robot Localization Using Sonar", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-9, No. 2, pp. 325-332, March 1987.
18. Moravec, H.P., Alberto, E., "High Resolution Maps from Wide Angle Sonar", 1985 *IEEE*, pp. 116-121.
19. Beckerman, M., Oblow, E.M., "Treatment of Systematic Errors in the Processing of Wide Angle Sonar Sensor Data for Robotic Navigation", *ORNL/TM-10763, CESAR-88/07*, April 1988.
20. Crowley, J.L., "Dynamic World Modeling for an Intelligent Mobile Robot Using a Rotating Ultra-Sonic Ranging Device", 1985 *IEEE*, pp.128-135.
21. Burks, B.L., deSaussure, G., Weisbin, C.R., Jones, J.P., Hamel, W.R., "Autonomous Navigation, Exploration, And Recognition Using the HERMIES-IIB Robot", *IEEE EXPERT*, Winter 1987, pp 18-27.
22. Marioli, D., Sardini, E., Taroni, A., "Shape Determination and Robot Arm Control Positioning by Means of Ultrasonics", *Proceedings of the 7th International Conference on Robot Vision and Sensory Controls*, pp. 171-182, February 2-4, 1988, Zurich, Switzerland.
23. Warnecke, H.J., Langen, A., "New Ultrasonic Sensors for Robotic Application Based on Beam Forming", *Proceedings of the 7th International Conference on Robot Vision and Sensory Controls*, pp. 149-160., February 2-4, 1988, Zurich, Switzerland.
24. Rock, D., Redus, J., Marr, L., Gohlke, M., Hartnett, D., "Falcon Eye Forward-Looking Infrared (FLIR) System", *Proceedings of SPIE-International Society for Optical Engineering*, Volume 782, pp. 38-45, May 19-21, 1987, Orlando, Florida.
25. Pellegrini, P.W., Golubovic, A., Ludington, C.E., "A Comparison of Iridium Silicide and Platinaum Silicide Photodiodes", *Proceedings of SPIE-International Society for Optical Engineering*, Volume 782, pp. 93-98, May 19-21, 1987, Orlando, Florida.
26. Mooney, J.M., Silverman, J., Weeks, M.M., "PtSi Internal Photoemission: Theory and Experiment", *Proceedings for SPIE-International Society for Optical Engineering*, Volume 782, pp. 99-107, May 19-21, 1987, Orlando, Florida.
27. Aguilera, R., "256x256 Hybrid Schottky Focal Plane Arrays", *Proceedings of SPIE-International Society for Optical Engineering*, Volume 782, pp. 108-113, May 19-21, 1987, Orlando, Florida.
28. Kosonocky, W.F., Hughes, G.W., "High Fill Factor Silicide Monolithic Arrays", *Proceedings of SPIE-International Society for Optical Engineering*, Volume 782, pp. 114-120, May 19-21, 1987, Orlando, Florida.
29. Murguia, J.E., Ewing, W.S., "Statistical Characterization of a Large PtSi Focal Plane Array",

- Proceedings of SPIE-International Society for Optical Engineering, Volume 782, pp. 121-127, May 19-21, 1987, Orlando, Florida.
30. J. P. Karlen, J. M. Thompson, J. D. Farrell, "Design and Control of Modular, Kinematically Redundant Manipulators", Second AIAA / NASA / USAF Symposium on Automation, Robotics and Advanced Computing for the National Space Program, March 9-11, 1987, Arlington, VA (Robotics Research Corporation).
  31. J. D. Farrell, "Pragmatic Control of Kinematically Redundant Manipulators", 1988 American Control Conference, July 15-17, 1988, Atlanta, GA (Robotics Research Corporation).
  32. J. S. Albus, H. G. McCain, R. Lumia, "NASA / NBS Standard Reference Model for Telerobot Control System Architecture (NASREM)", National Bureau of Standards, Technical Note 1235, June, 1987.



# Preliminary Study of a Serial-Parallel Redundant Manipulator

Vincent Hayward  
Ronald Kurtz

McGill Research Center for Intelligent Machines  
3480 University Street, Montréal, Québec Canada H3A 2A7

## Abstract

The manipulator design discussed here results from the examination of some of the reasons why redundancy is necessary in general purpose manipulation systems. A spherical joint design actuated "in-parallel", having the many advantages of parallel actuation, is described. In addition, the benefits of using redundant actuators are discussed and illustrated in our design by the elimination of loci of singularities from the usable workspace with the addition of only one actuator. Finally, what is known by the authors about space robotics requirements is summarized and the relevance of the proposed design matched against these requirements. The design problems outlined in this paper are viewed as much from the mechanical engineering aspect as from concerns arising from the control and the programming of manipulators.

## 1 Introduction

In general, design, seen as a problem solving activity, is very unconstrained. It has been observed that design is less a goal-driven activity than a process-driven activity: the design 'process' is picked by the designer according to a complex set of reasons.<sup>1</sup> In the case of manipulators, only a surprisingly small number of design processes have been utilized by the industry, resulting in a small number of design styles. In the recent years, a greater amount of manipulator design problems have been tackled in research laboratories.

Optimality is a notion which is difficult to incorporate in the design activity, because optimality entails the existence of a well defined objective function. In design, it is difficult to define such a function since the space over which this function would be defined cannot be known before the end-result of the design process has been satisfactorily described. Nonetheless, a design can be declared optimal with respect to a particular model and particular criteria defined over the variables of this model. The relevance of the model is then of course an essential question.

Design occurs by satisfying an open set of constraints resulting in part from the laws of nature, some of which in the case of manipulators are captured by the equations of kinematics and dynamics. Kinematics and dynamics have little synthetic power: they only permit a designer to improve a proposed design through analysis or optimization. However, qualitative explorations seem possible as demonstrated by Salisbury in the context of whole arm manipulation.<sup>2</sup>

Other constraints result from technological feasibility. These are of course difficult to obtain since they depend on the accuracy of available information, the risk involved in creating new technologies, and the rate of improvement.

The remainder of the constraints encompasses a set of desired properties which can be quite arbitrary. These are decided upon by the designer for reasons that have to do with culture, tradition, personality, wit, corporate image, budget, trends, fashion, and so on.

As a result, a design goal often cannot be formalized; instead, as commented above, a generative method is selected. Possibilities are matched against the criteria that have been decided upon before hand. Unpromising alternatives of the successive versions are filtered in a process which is reminiscent of a technique known in artificial intelligence as "means-end analysis." The definition of quantitative criteria may help to automate part of the search process. The final goal is known once successive generations have been filtered by the constraints. For example, the approach elaborated by D. Tesar for the design of manipulators, employs a selection method based on a hierarchy of criteria.<sup>3</sup> However, it is unlikely that the design process can ever be reduced solely to an explicit search process.

The most common methodology first entails the creation of generic modules which can be instantiated into a collection of devices having scaled properties (size, power and so on). The advantages of such an approach are well known and discussed at length in computer science literature. The principles put forward in computer science are standardization (interface rules), polymorphism (hiding implementation), and composition (larger blocks made of smaller ones). They promote abstractions, reliability, ease of maintenance, and top-down design. These principles clearly apply a great deal to electro-mechanical design as well. The second part of this methodology is to decide upon a framework structure, which describes how modules relate to each other. In order to deal with complexity, hierarchical organizations are predominantly proposed. However, a number of other alternatives are also available.

## 2 Goals

Vastly different 'designer goals' can be noticed in discussions pertaining to robotic end-effector designs, from "Nature produces systems which utilize real hardware that operates according to physical principles...the intent [of the design] is not to imply that the development of such systems will be an easy task, only that such systems can be developed",<sup>4</sup> to "we feel that what is needed is a *medium-complexity* end effector: a device that combines the ease of control characteristic of the simple grippers with some of the versatility of the complex hands."<sup>5</sup> In the case of walking machines, other motivations are sometimes invoked, for example in the following proposal: "Among the animals that one might wish to emulate, an obvious class is that of the dinosaur."<sup>6</sup> The list of justifications given by the author are no less convincing than those given in the other references.

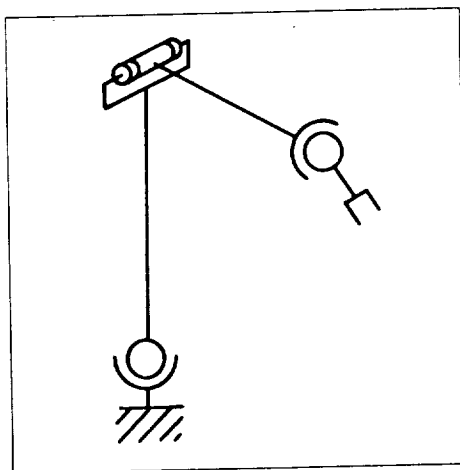
In our case, an exploratory study of redundancy was our motivating factor for the arm design. It has been previously recognized that redundancy is not only desirable, but necessary to the design of general purpose manipulators.<sup>7</sup> From this initial premise, a set of thirty reasons why redundancy is useful are exhibited. Resulting from this discussion, a mixed serial-parallel kinematic structure has been proposed.

Parallel designs, because of their possibility to achieve low inertia and structural rigidity, are very appealing. Unfortunately, the theory of mechanisms shows that the workspace is generally limited. Hence, the structure we proposed is a hybrid structure, designed to allow

a trade-off between conflicting requirements. It has the following properties:

1. Hand motion decoupled from that of major links to augment ability to conform to obstacles achieved by redundancy.
2. Limited seriality.
3. Parallel actuation to achieve high bandwidth and rigidity.
4. A truss assembly can be devised to achieve rapid impact transient damping and good load/weight ratio.
5. Possibility to de-locate actuators through tendon motion transmission.
6. Workspace augmentation and backlash elimination achieved through actuator redundancy.

The proposed design (see figure 1) consists of a spherical wrist and a shoulder joint with an interposed revolute elbow joint. We see that a compact spherical element with a large range of motion and sound mechanical design is essential. This can be achieved through in-parallel actuation with actuator redundancy.



**Figure 1.** Spherical joints are actuated "in-parallel."

From the general case of a fully parallel wrist an particular arrangement has been derived (see 2), and its models written.<sup>8</sup>

The results of this study are presented in the following subsections.

### **3 Parallel Wrist Properties**

#### **3.1 Workspace**

Assuming that the geometry of the mechanism can be represented in terms of cylinders, the interference of all moving parts can be analytically derived. The following plot (figure 3) depicts the range of swivel  $\theta$  for each value of  $\psi$  and  $\phi$ .  $\theta$ ,  $\psi$  and  $\phi$  are three Euler angles where  $\psi$  is a rotation about the x-axis,  $\phi$  is a rotation about the new y-axis, and  $\theta$  is a rotation about the new z-axis.

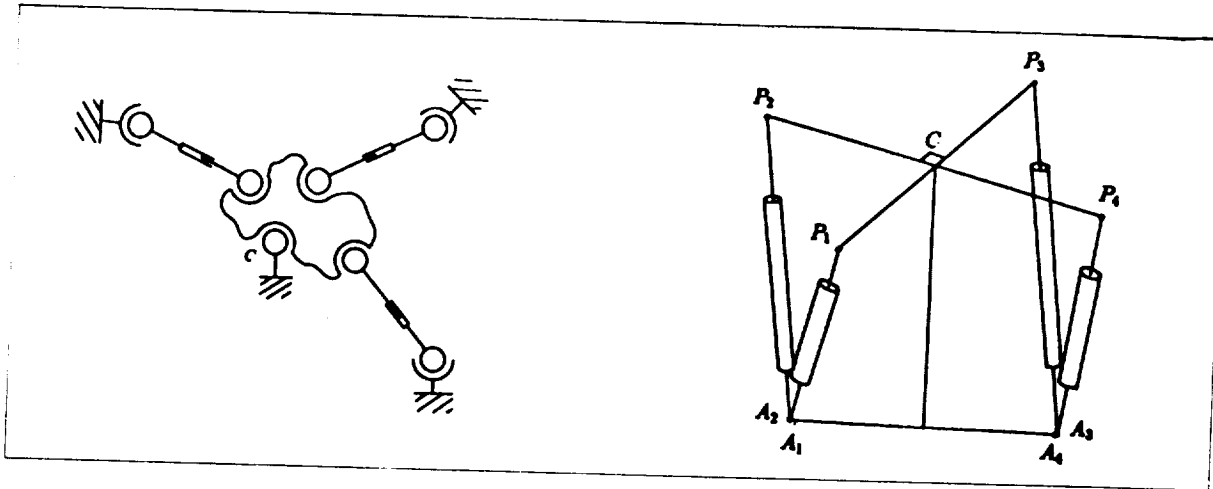


Figure 2.

Left: General case of a fully parallel wrist; Right: Practical proposed redundant mechanism.

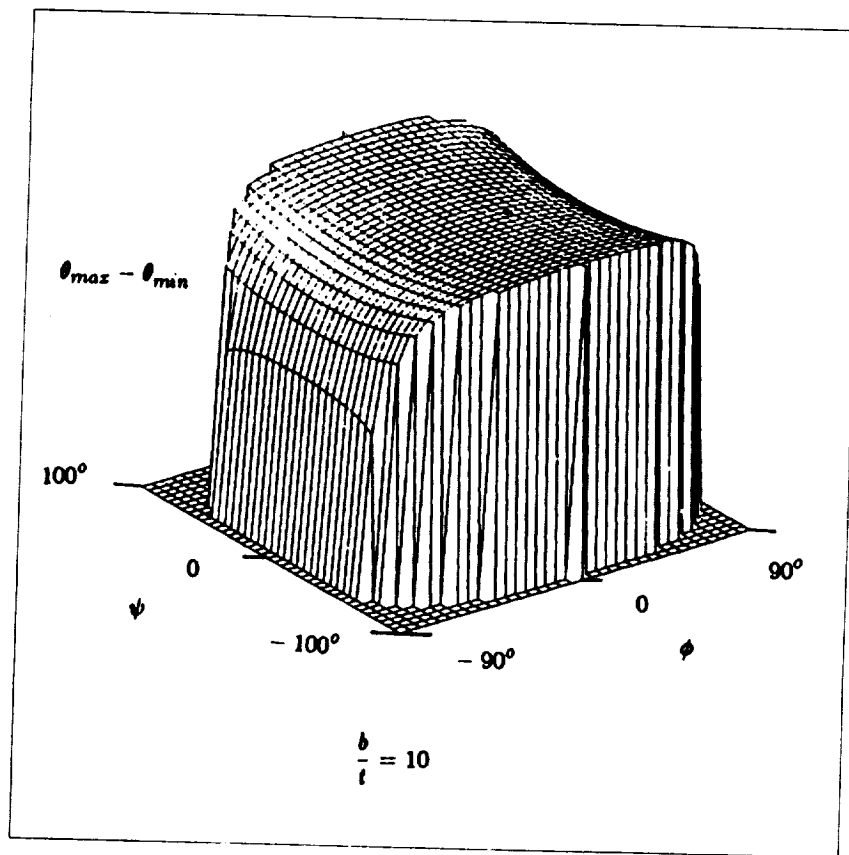


Figure 3.

Workspace with a length to thickness ratio of 10. The dependency of variations of  $\theta$  is plotted against those of  $\phi$  and  $\psi$ .



### 3.2 Kinematic Equations and Jacobians

The inverse and forward kinematic models can easily be derived in analytic form, as well as the forward and inverse Jacobian matrices.

### 3.3 Singularities

A remarkable feature resulting from the addition of a fourth actuator is the elimination of the loci of singularities. It is in fact possible to show that for the grouped actuator case, all singularities are eliminated except when the plane containing the points  $P_i$ 's also contains the  $A_i$ 's. This configuration is in fact outside the usable workspace of the manipulator as previously defined.

### 3.4 Dexterity

We have analyzed the dexterity of the parallel redundant wrist by looking at the Jacobian condition number  $k(J)$ . The condition number can be physically interpreted as the amplification of round off error when going from input to output coordinates, and hence is a direct measure of the accuracy of the wrist in a specific configuration. The condition number ranges in value from one (isotropy) to infinity (singularity) and thus can be used as a measure of the "distance" the particular wrist configuration is from a singularity. The condition number is given by:<sup>9</sup>

$$k(J) \equiv \|J\| \|J^{-1}\|$$

where we can use the frame invariant Frobenius norm with weighting matrix  $W$ :

$$\|J\|_F \equiv \sqrt{\text{tr}(J^T W J)}$$

For a redundant manipulator  $J$  is non-square, and hence the condition number is defined as the maximum singular value of  $JJ^T$  divided by the minimum singular value.

Figure 4 plots the dexterity (defined as  $D = 1/k(J)$ ) of the grouped actuator wrist against three Euler angles. It is interesting to note that there are several configurations where the wrist is isotropic ( $D = 1$ ), providing good operating points for fine and accurate motions. As the tilt angle  $\phi$  increases there is a general loss of dexterity, culminating in the singularity ( $D = 0$ ) at  $\phi = \pm 90^\circ$ . Large values of  $\phi$  lie outside the workspace, so the poor dexterity at these points can be ignored. For this design the dexterity is high in the range:

$$-60^\circ \leq \phi \leq 60^\circ, \quad -90^\circ \leq \psi \leq 90^\circ, \quad -135^\circ \leq \theta \leq 135^\circ$$

This provides a large usable workspace free of singularities and well suited for accurate motions.

## 4 Inclusion Into An Arm Design

Once the kinematic feasibility has been shown, the next step is the inclusion of the spherical assembly into a truss structure. The figure 5 shows one possibility using rather simple technology.

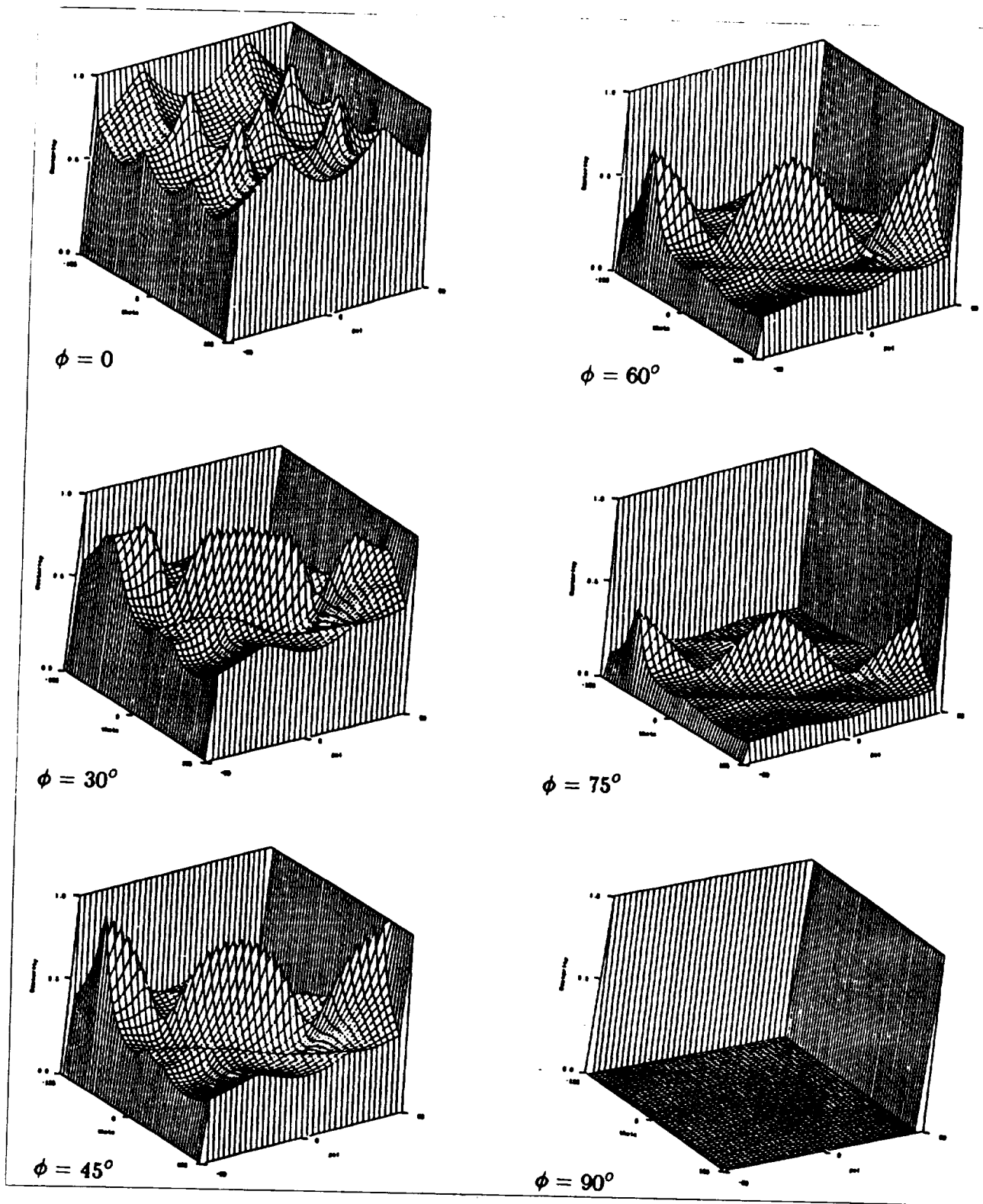
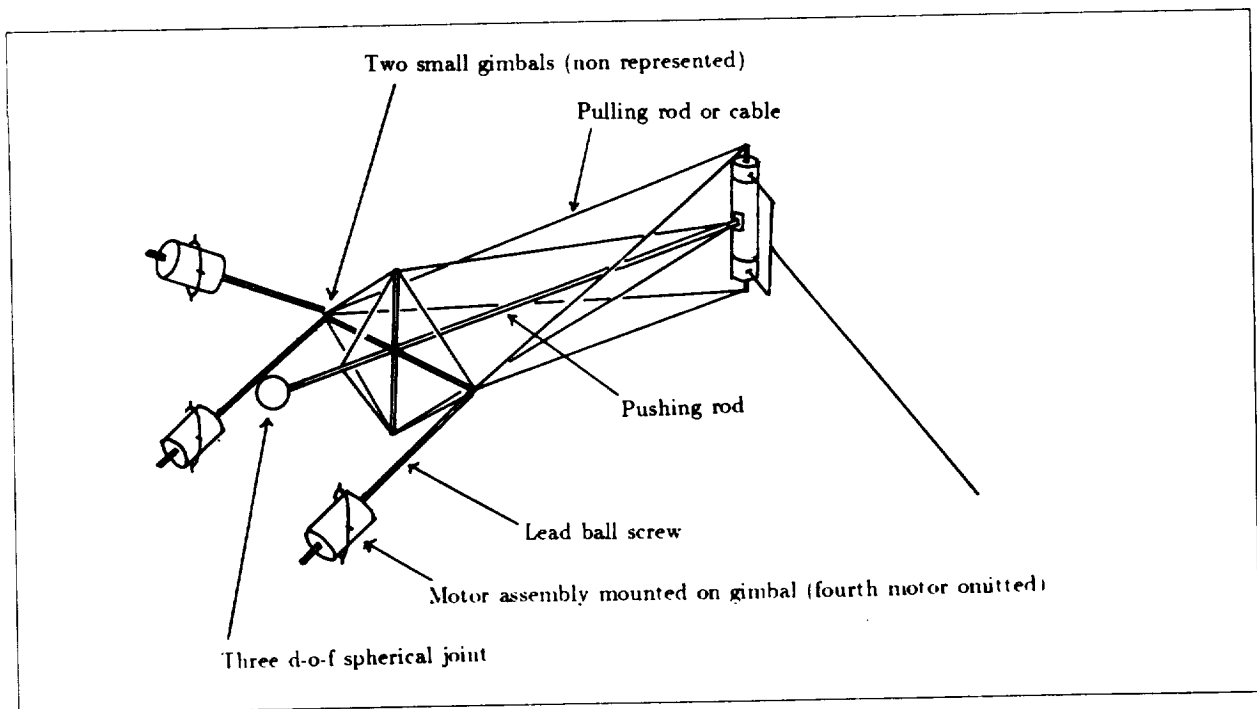


Figure 4.

Dexterity plotted for all lengths set to 1 versus angles  $\theta$  and  $\psi$ . Each plot is for a different value of  $\phi$ . The isotropic points ( $D = 1$ ) are present for  $\phi = 0$  and  $\phi = 45^\circ$ . Only when  $\phi = 90^\circ$  is the manipulator singular and the dexterity identically zero.



**Figure 5.**

Truss assembly of the proximal link with integrated parallel actuation. Note the de-located actuators.

The figure does not show how a wrist can be integrated. At the present, we are investigating the possibility of a tendon-driven spherical parallel mechanism which has identical properties as when dual action actuators are used.

Several remarks can be made about this design:

- *Skeletons*: Limbs in nature come in two varieties: endo-skeletons and exo-skeletons. So far, the design of artificial manipulators has followed a similar categorization (linear actuators: exo-skeletons, rotary actuators: endo-skeletons). Clearly the proposed design falls in the endo-skeleton category with the material used in compression located *inside* the material used in extension.
- *Actuator and Sensor Integration*: The truss design offers the advantage of making actuators and sensors an integral part of the structure, thus resulting in an economy of means.
- *Modularity*: The elements that make up such a design fall into a very small number of categories which facilitate design and construction. These are:
  1. Linear actuator, preferably slender, light and back-drivable.
  2. Pushing rods. From the load requirements, structural mechanics will tell the desired characteristics.
  3. Pulling rods. Same as above.
  4. Universal joint. Same as above.

5. Spherical joint. Same as above. An attractive possibility is a true ball-and socket assembly.
6. Multiway rigid connection for rods.

## 5 Relevance to Space Applications

In addition to the mobility criteria which have guided our choices through-out this discussion, a few additional points could be made with respect to space requirements.

- *High-reliability*: Space hardware has a mandate for reliability. The modular design outlined above can only help reliability. In addition, the actuator redundancy preserves some of the maneuverability in case of failure of one actuator.
- *Weight*: This issue is of course very well addressed by our proposal.
- *Power Consumption*: This requirement must be satisfied by an appropriate motor-reductor technology independent from this particular proposal.
- *Lubrication*: Same as above.
- *Back-drivability*: Same as above.
- *Temperature gradient* The deformation of structures under temperature gradient can be measured and compensated for. In fact, an arm made of a struss structure offers quite interesting possibilities. For example, the temperature of the rods can be measured and deformation computed from this information.
- *Control*: All the kinematic models are easily obtained in closed form. The control of the kinematic redundancy can easily be performed because of the decoupling of the arm self-motion from the hand motion. The dynamic model can be derived very simply because of the various decouplings. The structure can be tuned to absorb impact transients which improves the frequency response.

## 6 Conclusion

A number of issues remain to be addressed before such a proposal could reach the stage of implementation: choice of sensors, motors, and so on. However, kinematic feasibility has been established and a sound structural design is easy to obtain. Actuation redundancy also lead to interesting control issues.

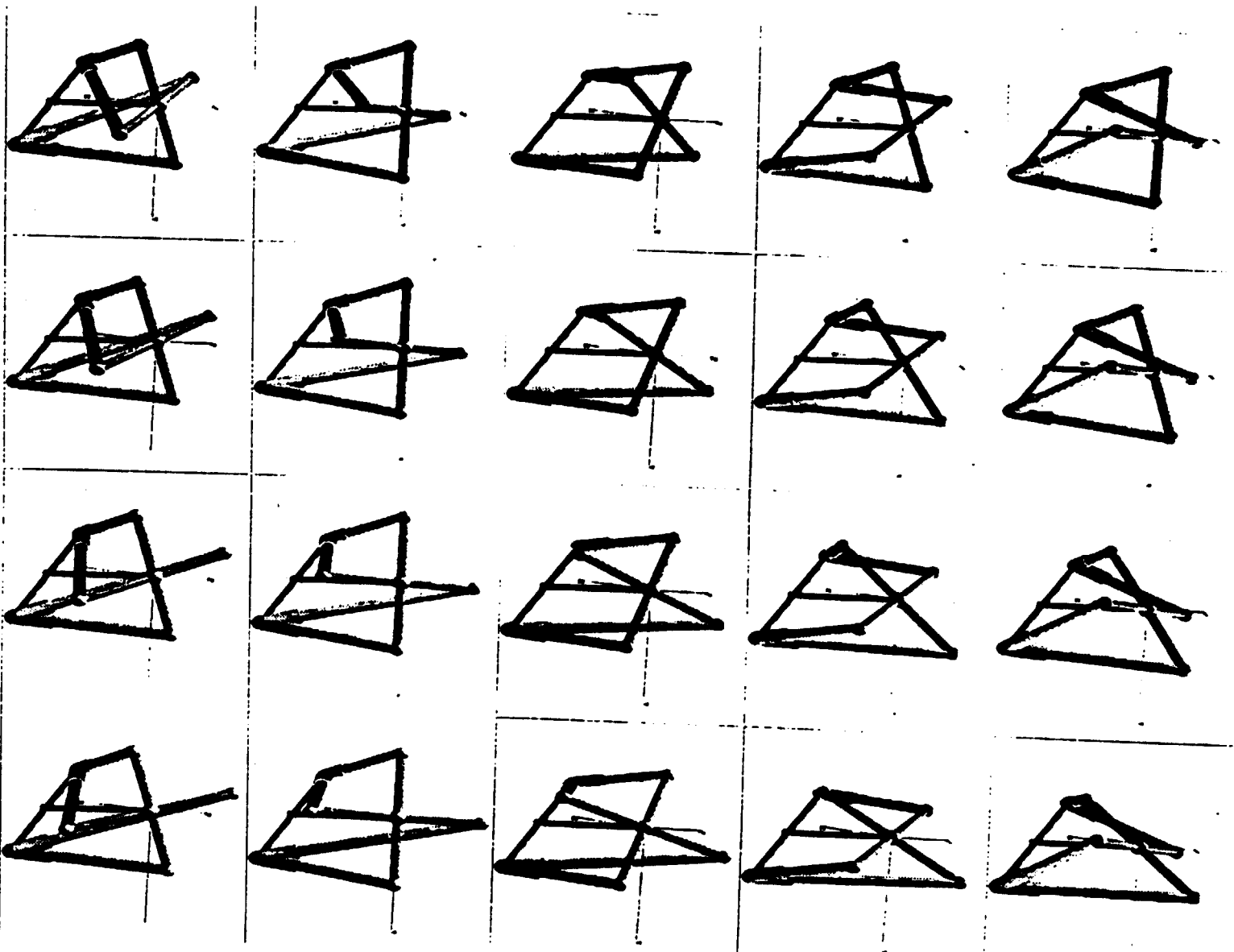
## 7 Acknowledgement

Many thanks to Stéphane Aubry who helped solve the kinematics. Ajit Nilakantan and Faycal Kahloun (both from Cimmetry Inc.) did the CAD modeling and contributed numerous ingenious suggestions.

The work was made possible by funding provided by NSERC the Natural Sciences and Engineering Research Council of Canada, and FCAR "Fonds pour la Formation des Chercheurs et l'Aide à la Recherche," Québec.

## 8 References

1. Simon, H. A. 1985. *The sciences of artificial intelligence*, MIT Press.
2. Salisbury, K. 1987. Whole arm manipulation. In *Fourth Int. Symposium on Robotics Research*. R. C. Bolles and B. Roth (Eds.), MIT Press
3. Tesar, D., Cleary, K. 1989. Decision making criteria for redundant manipulator. In *Robots with redundancy: design, sensing and control*, NATO Series, A. Bejczy (Ed.), Springer Verlag, in press.
4. Jacobsen, S. C., Iversen, E. K., Knutti, D. F., Johnson, R. T., Biggers, K. B. 1986 (San-Francisco, Ca). Design of the UTAH/MIT dextrous hand. *IEEE Conf. Robotics and Automation*.
5. Ulrich, N., Paul, R.P., Bajczy. R. 1988 (Philadelphia, Pa, April). A medium-complexity compliant end effector. *IEEE Conf. Robotics and Automation*.
6. Todd, D. J. 1988 (October, Manchester, UK). Stability in Four-legged walking vehicles. *The second workshop on manipulators, sensors and steps toward mobility*.
7. Hayward V. 1989. An analysis of redundant manipulators from several view-points. In *Robots with redundancy: design, sensing and control*, NATO Series, A. Bejczy (Ed.), Springer Verlag, in press.
8. Hayward, V., Kurtz, R. 1989. Modeling of a parallel wrist mechanism with actuator redundancy. *Technical Report, McGill Research Center for Intelligent Machines*.
9. Angeles, J., 1988. Isotropy criteria in the kinematic design and control of redundant manipulators. *Technical Report, McGill Research Center for Intelligent Machines*.



## **TELEOPERATION 1**





## The JPL Telerobot Operator Control Station: Part I - Hardware

Edwin P. Kan\*

John T. Tower, George W. Hunka, Glenn J. VanSant\*\*

### ABSTRACT

The Operator Control Station of the JPL/NASA Telerobot Demonstrator System provides the man-machine interface between the operator and the System. It provides all the hardware and software for accepting human input for the direct and indirect (supervised) manipulation of the robot arms and tools for task execution. Hardware and software are also provided for the display and feedback of information and control data for the operator's consumption and interaction with the task being executed. This paper, Part I, addresses the hardware design, system architecture, its integration and interface with the rest of the Telerobot Demonstrator System.

### 1.0 INTRODUCTION

The JPL/NASA (Jet Propulsion Laboratory / National Aeronautics and Space Administration) Telerobot Demonstrator System is a research testbed for the development, integration and testing of advanced robot control technologies [ref.1]. The component technologies and system-wide design experiences derived from such a system development and technology demonstration are targeted for use in future space programs, including the NASA's Flight Telerobot Servicer project [2].

Being a complex system involving many disciplines and technologies, the Telerobot Demonstrator System is designed as a hierarchical system, consisting of an Operator Control Station (OCS), a Reasoning and Planning Subsystem (TPR, also known as the Artificial Intelligence Planner, AIP), a Run-Time Control Subsystem (RTC), a Manipulator Control and Mechanization Subsystem (MCM), a Sensing and Perception Subsystem (S&P), and a System Executive (SE) Subsystem. Implicit in this architecture is the Human Operator, who is the 'commander' of the System, located within the OCS.

This telerobot system is a hybrid between teleoperated and robotic (autonomous) system. It is designed to operate where either pure teleoperated or pure autonomous operation is too complex, too inefficient or simply infeasible. Design performance goals for a telerobot system are primarily targeted for space applications in construction, assembly/disassembly, and servicing/maintenance. Telerobotic capabilities are to be demonstrated by performing laboratory tasks simulating those encountered in servicing satellites in orbit. Typically, they include coordinated two-arm manipulation of a large module, an ORU (Orbit Replacement Unit), and include the grappling/halting of a rotating satellite. Dexterous operations in terms of removal of panels, bolts, electrical connectors, tool exchange, object manipulation with precisely defined or loosely defined data bases, are in the list of demonstrations.

Special hardware and software have to be designed into the OCS. It contains state-of-the-art hardware, both mechanical and computing, for providing control input to the System. It contains software, in controls as well as human operator interface, for real-time and user-friendly interaction. Video displays for text, graphics and camera images are provided for operator consumption; where appropriate, voice input/output is provided to reduce operator work-load. Data manipulation such as object designation capability is provided for efficient task definition and execution. Access to all Telerobot subsystems is provided for software development and on-line monitoring.

There is a critical need for efficient and effective interaction between the Operator and the System. The hybrid characteristics of this telerobot system are such that teleoperated control and autonomous control are frequently traded and/or shared, in a continuum fashion. This mode of control is sometimes referred to as 'supervised control'. Thus, human factors issues are very demanding in the design of the OCS.

This paper, Part I, addresses the hardware design, system architecture, and its integration and interface with the rest of the Telerobot Demonstrator System. Description of the software is included in Part II of this paper [3].

\* Jet Propulsion Laboratory, California Institute of Technology, Pasadena, Ca.

\*\* GE Aerospace (formerly RCA) / Advanced Technology Laboratory, Moorestown, N.J.

## 2.0 OPERATOR CONTROL STATION (OCS)

Figure 1 is the functional block diagram of the OCS, showing all its functional components and its interface with the other subsystems, namely the S&P, TELEOP, TPR (AIP), SE, RTC and MCM.

Implicit in the OCS is the human Operator, who uses the OCS to command and interface with the System. The Operator manages system configuration, transmits system information and receives feedback from the System. The OCS provides capability for the Operator to coordinate and monitor all other subsystems, permits the Operator to direct, supervise, and execute robotic and teleoperation control. In the JPL Telerobot Demonstrator System, OCS is designed for two operators, the Main Operator and Auxiliary Operator (also known as the Test Conductor). The Main Operator has the capability to execute all functions regardless of the absence or presence of the Auxiliary Operator.

The OCS hardware is a station, in a 'controlled' room environment where lighting, sound and sight are controllable, and stations the Main Operator and Auxiliary Operator. The station is equipped with multiple monitors for video and graphics displays and mixing. Audio and voice input/output systems are provided for operator command inputs in addition to keyboard inputs. Mechanical input devices for teleoperation and shared robotic/teleoperation control are provided. Multiple processors and computer networking is provided for OCS functions, planning functions and system management functions. And, the OCS is designed to ergonomic guidelines and standards.

Interface to the other subsystems is mainly via the ethernet network. A custom Network Interface Package (NIP) [ref. 4] software has been developed to standardize NIP network communication transactions between the different subsystems. These transactions can include commands, macros, statuses, messages, and combinations thereof. In such a way, the subsystems can be kept independent and cleanly separated from one another, as dictated in the hierarchical architecture.

In addition to this NIP interface, the Operator can log on to all the subsystems via the OCS computer terminal, via a terminal emulation mode. This is foreseen to be mostly used in system debugging, system setup, and certain detailed analysis operations, rather than for normal telerobot operations during task execution.

The present OCS design is based on experiences and evaluation of different systems, including the Los Alamos Scientific Laboratory, Fermi National Laboratory, Argonne National Laboratory, Oakridge National Laboratory M-2 and ASM systems [ref.5,6], and in-house research work. Human engineering factors are considered in this design [ref.7]. Other functional specifications and interface designs are based on the performance goals of the telerobot system and the architecture/data flow of the entire System. For further details on the functional requirements of the System and of the OCS, refer to [ref. 8,9].

Before describing the hardware design details, it is to be noted that the present OCS design is an evolutionary design which will evolve and change as the telerobot technology matures, both in system design and in component design. The present design is believed to have the necessary 'hooks and scars' for future system expansion. Despite its flexibility, certain architectural features are recognized to be suboptimal because of project constraints. Among the suboptimal design features is the sharing of certain OCS and TPR functions, even though they are distinctly separate subsystems. The Operator now inputs to the System via the OCS terminal as well as via the TPR terminal. The TPR high resolution graphics terminal is actually located at the OCS, used for operator input during autonomous and supervised control operations. This input process is independent of the use of the OCS SUN computer terminal during teleoperation and system mode operations. Otherwise, the OCS communicates with the TPR in the same fashion as the other subsystems, i.e. through the ethernet network using the NIP. One future approach to unify the design is to subsume the TPR functions into the OCS, at least in the operator interface features; in that case, the Operator will have only one terminal to interface with all the subsystems.

## 3.0 THE OCS HARDWARE

The physical layout of the OCS is shown in Figures 2 and 3. The OCS has an L-configuration, composed of six racks. Racks A, B, C, and D constitute the Main Operator station. The remaining racks E and F constitute the Auxiliary Operator station. The main station has all the controls necessary for the primary Operator to operate and execute telerobotic operations alone and independent of the auxiliary station. In fact, the auxiliary station has only a subset of the capabilities at the main station, to be used primarily by a Test Conductor-type secondary operator for monitoring purposes.

The primary station houses the right and left Force Reflecting Hand Controllers (FRHC) and their electronics, and the video monitors that the operation requires for teleoperation [ref.10]. The station also provides a keyboard for programmed control and user interface to the OCS computer, and to the other telerobot subsystems. The main Operator also has the OCS primary computer monitor and the TPR computer monitor in front of him for man-machine interface. For future expansion, a slot has been allocated in this main station for a real-time high-speed graphics machine, such as the Silicon Graphics IRIS system. At the present configuration, this station has two other graphics processes: (i) force-torque sensor displays, from the left robot sensor and from the right robot sensor; (ii) graphic overlays for the "object designation" process (see Section 3.5.3). The Main Operator has control over a voice recognition and synthesizer system, while he wears a head-gear consisting of a microphone and earphone. Direct commands can be issued by voice as well as through the keyboard, while more commands and parameter specifications can be input to the System via menu input processes at the OCS computer terminal.

The secondary station houses two additional video monitors and the OCS secondary computer monitor. While the secondary Operator does not have any voice input capability, he can always enter all the commands via the OCS secondary computer terminal, via direct command inputs using the keyboard or via the menu selection process, displayed to him on the OCS secondary computer monitor. Here, the Operator cannot provide the teleoperation inputs because of the absence of the FRHC's. All graphics, overlays and video images can be displayed to his two monitors, as routed by an OCS process of video switching.

Common to both stations are the video switcher, which is now configured to route a maximum of 16 RGB color channels to a maximum of 16 RGB color output monitors. Multiple views of the same input channel are possible with the present switcher. Other audio mixing, amplifier, video recording equipments are installed for the use by both Operators. Both stations have their own individual emergency kill button, which can also be used for a special halt-retract function.

The OCS computer is configured by a SUN 3/160 workstation as the primary computer, and with a SUN 3/60 workstation as the secondary computer. Each has a monochrome display monitor. The SUN system was selected for many reasons, including its wide-area ethernet networking capability, efficient and extensive software development facilities, and general compatibility with the development environment and computers used by the other telerobot subsystems. The SUN View and SUN Tool facilities provide user friendly processes, including multi-window and multi-process capabilities that provide versatile terminal emulation communication between the OCS and the other subsystems.

### **3.1 The Force Reflecting Hand Controller (FRHC) Workstation**

The central input device to the primary Operator is the set of two FRHC's, one right-handed and the other left-handed. (See Figure 8.) The Operator manipulates the controllers seated at the center of the workstation, while observing the arm responses on monitors providing multiple views of the robot work volume. As a safety and training measure, a direct viewing window, shown in the floor view diagram (Figure 2), is provided into the arm work area to verify his operations and to insure that the area is clear of personnel.

Mounting bases of the FRHC's can be adjusted to allow the hand grips of the controllers to be placed one above the other and rotated in a one foot circle. Figure 4 shows the range of adjustment for the FRHCs. Adjustment mechanisms can easily be reached and operated by the operator while seated.

### **3.2 The Operator Workstation Video Graphics**

#### **3.2.1 Wing and Overhead Monitors**

To provide the operator with the necessary perspective to effectively and safely operate in the robot work space, multiple views are provided by two cameras located at either side of the work space (wing cameras), and one centered, forward looking camera somewhat above the work space (overhead camera). These views will generally be global, presenting the operator with an overall picture of the work space from different perspectives. The OCS also provides a stereo view from a camera pair mounted on a third robotic arm which can be positioned by the operator for an optimum view. Past experience on teleoperation using stereo views has shown that stereo vision provides additional depth cueing necessary to perform tasks efficiently.

### 3.2.2 Stereovision

The stereovision system, centrally located at the primary station, is a completely passive design. It has the advantage of not requiring any mechanism for synchronized switching to properly view the stereo images, and can maintain the stereo effect over a relatively wide latitude of head motion. Two orthogonally positioned monitors with cross-polarized face plates display views from the stereo pair cameras. The monitor views are superimposed onto a 45-degree beamsplitting screen having a 50% transmission/reflection ratio. Located at eye-level directly in front of the operator, images from the screen are directed to the appropriate eye by a pair of polarized glasses worn by the operator. Figure 5 shows the configuration and adjustment mounts of the stereo system. The left parallax is given by the bottom vertically positioned monitor, the right parallax by the horizontal monitor. To have the lines of the coincident images scanned in the same direction, the lower monitor is modified to have a reversed horizontal scan. The alignment of the images on the beamsplitter is critical to performance and positioning adjustments are provided. While the horizontal monitor is rigidly mounted within the OCS console, the vertically oriented monitor can be adjusted in many directions for precise alignment. The beamsplitter is also angularly adjustable. The adjustment ranges are indicated in the following table:

Stereo Vision Mount:	X	±1.50 in.
Range of Adjustments:	Y	±0.50 in.
(refer to Figure 5)	Z	±1.50 in.
	YR	±10°
	ZR	±10°
	Mirror	45°±5°
	Mirror X	±0.50 in.

The mirror is also removable, so that the Operator can use the 'right perspective' (i.e. the top horizontally mounted) monitor by itself; in this case, the Operator no longer needs to wear the polarized glasses.

### 3.2.3 Force/Torque Graphics

Information on the forces/torques experienced by the manipulators is useful in performing telerobot tasks with or without force reflection. Force/torque information is highly desirable to assure that excessive forces are not imparted to the work object in all modes of both teleoperations and programmed control. Wrist force/torque sensors provide this information. The OCS is equipped with two force/torque monitors which graphically display applied forces in the six-axes of each wrist in a readily interpreted manner. These monitors are placed side-by-side and are conveniently located directly above the stereo window.

### 3.2.4 OCS Monitors

The OCS monitor provides the user interface to the OCS software. The OCS user interface consists of a variety of predefined windows, pop-up menus, graphic buttons, and panels that are configured to provide a consistent command and message environment for the OCS operator. For details, refer to [ref.3]. The following table documents the full complement of monitors provided within the operator workstation:

<u>Operator Station Monitors</u>			
	<i>Designation</i>	<i>Type</i>	<i>Manufacturer/Model#</i>
<i>Primary Station:</i>	Left Wing	19" RGB	Barco CD531
	Right Wing	19" RGB	Barco CD531
	Overhead	19" RGB	Barco CD531
	Left Stereo	19" RGB	Barco CDCT6351
	Right Stereo	19" RGB	Barco CDCT6351
	Left F/T	10" RGB	Barco MCD10B
	Right F/T	10" RGB	Barco MCD10B
	Primary OCS	19" Mono	Sun (3/160)
	TPR terminal	19" Mono	Symbolics (3640)
<i>Secondary Station:</i>	Secondary OCS	19" Mono	Sun (3/60)
	Aux. monitor (2)	13" RGB	Barco CD233

### 3.2.5 Graphic Generators

Two New Media Graphics 9000 graphic generators are installed in Rack E. These units permit color alphanumerics to be inserted onto selected video. They also serve in the object designation mode to generate wire-frame outlines of objects contained in a world model data base (see section 3.5.3). Using a mouse to designate vertices of identifiable objects contained in a video scene, the graphics generator output is superimposed on the video scene, automatically establishing correct relative orientations to redefine or update the world model. Future expansion to more than two graphic generators is foreseen; spare slots on the racks are allocated for such expansion.

### 3.3 OCS Video Processing Elements

Much of the OCS rack space is devoted to video system requirements which include not only display monitors, but also a video router and video encoders and decoders.

Some clarification may be given to nomenclatures of the station monitors in prior discussions, since the video router allows any video to be displayed on any monitor. Suffice it to say that under normal operating conditions, an operator would use these nominal designations as described, i.e., the left wing camera would be displayed on the left wing monitor, etc., to maintain his correct orientation with the robotic work space.

The OCS video monitors, with noted exceptions, are all configured to accept Red-Green-Blue (RGB), video inputs. The video matrix switcher, supplied by BSM Systems, is capable of accepting and routing sixteen sets of RGB video signals to any or all of sixteen sets of output lines. Switcher control is via a RS-232 link between the OCS processor workstations and the switcher processor/controller. Configurational control of the switcher can be exercised either by keyboard or, as previously discussed, by voice command by the operator. The switcher will initially be configured to connect any of eleven inputs to any of twelve outputs, as shown in Figure 6, leaving provision for future expansion of the video system.

This figure also shows the RGB-to-NTSC encoding for input to the graphic overlay generators and VCR. While the graphic generators output RGB and can be routed directly to the switcher for monitor display, the VCR outputs NTSC video which must be decoded into RGB prior to display. A sync generator is included as part of the video subsystem, as shown, allowing OCS video to be slaved to the system master video clock. The encoders, decoders, and sync generator are Grass Valley Group components, and are also identified in Figure 6.

The video system described places at the disposal of the operator and test conductor complete and independent flexibility in selection of video source viewing monitors, graphics insertion capability, with provision for future system expansion.

### 3.4 Power Panel and Switches

#### 3.4.1 Station Control Panel

The OCS power control panel in Rack D contains the OCS rack power switches. Main power is supplied via a keyswitch, with panel indicators to verify that each rack is energized.

Three arm power switches are located on the panel for energizing the arms; switches for deactivating the arms in emergencies are independent assemblies which are discussed below. One of the assemblies is located on the primary station desk top convenient to the Main Operator, while the second unit is located at the secondary station convenient to the Auxiliary Operator.

Also, to eliminate excessive number of mouse(s) connected to the different computers, namely the OCS primary computer, the OCS secondary computer, the TPR computer, and the two graphics generators, the OCS primary computer mouse and the two graphics generator mouse(s) are combined into one. Software is designed in the OCS to share the use of one mouse for these three machines. Mouse jacks are also provided on the front panel for the detachment and attachment of the mouse(s).

#### 3.4.2 Emergency Stop Switch Assemblies

As a safety measure, the Main and Auxiliary Operators are each provided with a movable switch assembly which can abort arm activity in emergencies. These assemblies can be placed at any location convenient to the reach of the operator and the test observer. They are designed to be instantly recognized, having an indicator light, and have momentary push-to-stop mushroom-cap switches.

One more feature exists in the Main Operator's emergency stop switch assembly. It has the added switch for the selection of one of two available stop modes - (i) emergency stop mode, which immediately removes arm power and applies brakes; and (ii) a reflex stop mode, which causes the arms to retract to default locations out of the work zone. Selection of the reflex stop mode is indicated by the switch indicator lights.

### 3.5 Man-Machine Interfaces, Operator Aids, Human Factors

Space teleoperation places substantial demands on the human Operator. During the course of an exercise the Operator may be required to simultaneously:

- control, interpret, and respond to feedback from the remote manipulators;
- visually concentrate on task performance on various TV monitors;
- select and control camera positions for optimal views;
- interpret alphanumeric and graphic information which may be displayed;
- acknowledge and tend to visual and audible warnings.

To further tax the Operator, it may be necessary to continue these operations for extended periods of time. Under these conditions, the human factors issue becomes a critical area of research attention.

#### 3.5.1 Some General Design Features

The locations of controls and displays to be made available at the operator station and the auxiliary station were a subject of design study, and resulted in incorporating the following features in the design of the OCS, some of which were noted earlier:

- angled video monitors for viewing ease;
- primary head-on stereo display;
- adjustable arm controllers;
- contiguous left and right F/T displays;
- direct view window;
- relocatable, easily identified stop switches;
- stowable, adjustable position keyboards;
- desk-top workspace.

None of these general design features materially lessens the human stresses imposed during teleoperations. While they may add to comfort, convenience, and efficiency, they do not actively participate to reduce human stress. To reduce operator work load and attendant stress, operator aids have been incorporated in the OCS design.

#### 3.5.2 Operator Aids

##### 3.5.2.1 Voice Control Operator Aid

Experience has shown that substantial improvements in teleoperator performance are gained when voice commands are used to control auxiliary functions such as the camera system. During an exercise, the Operator's attention is expected to be focused on the control of the manipulators. Concentration on the task is particularly demanding when both arms are being used in a cooperative manner in manipulating a common work piece. Any distraction from the task should be prevented. It may be necessary for the Operator, however, to change camera angles as the work scene becomes obstructed or when objects leave the field-of-view. Voice control of cameras offers a natural, non-disruptive method for commanding the most useful views. It places minimal demands on the Operators' attention and allows the Operator to control the arms without interruption.

In the OCS, voice control is designed to control the camera pan and tilt motions; in this case, the stereo vision arm is placed on the 'vision arm'. Voice control is also extended to the control of the video router/switcher, allowing the operator to direct specific camera views for display on any desired station monitor. The video switcher, and video processing systems, have been described in Section 3.3.

Implementation of voice control requires a speech recognition system. Two competing speech recognition systems were considered for the OCS; Verbex S5000 and ITT 1280VME. Both are state-of-the-art, continuous speech, speaker dependent systems having efficient recognition algorithms which result in low error rates. The final selection is the Verbex S5000 system.

The Verbex S5000 system provides a >98% recognition accuracy and a very good (>95%) out-of-vocabulary rejection performance. The system can accommodate up to 80 active vocabulary words at any instance, and can include on-line storage of up to 500 vocabulary words, subsets of which can be swapped into active

memory rapidly (within one second). Careful pruning and partitioning of vocabulary words into subsets can provide error detection and correction strategies, which will further enhance the Type I and Type II accuracies of the recognition system. Other niceties of the system include well developed application development and user training software/procedures. The Verbex also provides on-board audio input/output capabilities that are needed to integrate the speech recognizer into the OCS racks. Lastly, for application development, i.e. pre-programming the vocabulary and grammar, an IBM PC is needed to run the Verbex high-level software. (Verbex is presently converting their software into VMS and UNIX host machines.) The OCS is designed to communicate with the Verbex via a serial RS-232 line.

### 3.5.2.2 Voice Synthesis as an Operator Aid

During operation, the telerobot Operator is barraged by visual information by way of camera video, graphics, and alphanumerics. While most information is supplied to the Operator via display monitors, certain types of information, such as status reports and warnings, can be more efficiently and more effectively conveyed by voice. A voice generator has also been shown to aid the Operator by providing operational cues in telerobotic systems which can share control between manual and supervisory modes.

A DECTalk DTC01-AA text-to-voice generator is used in the OCS as an operator aid. DECTalk is considered to be one of the most advanced speech generators commercially available. It can produce natural, human-quality voice messages with a vocabulary of greater than 20,000 words. In addition to this general common-word dictionary, DECTalk also contains a user-definable dictionary for specialized vocabularies used in specific applications. Standard ASCII messages from the computer workstation transmitted through an RS232 port are converted to voice messages at a controllable synthesizing output rate of 120 to 350 words per minute.

The OCS audio subsystem provides a standard commercial audio mixer/amplifier and speaker for combining audio sources from the speech synthesizer, microphone, and from audible warning devices.

### 3.5.3 Object Designation: Operator-Machine Interface

Developed for the Telerobot Demonstrator System is an interactive graphic overlay technique, better known as the "Object Designation and Verification" process [ref.10]. This process permits the Operator to interactively update the position and orientation data of known objects by a mouse point-and-designate sequence. Thus, any discrepancy, error, or unintentional displacement of objects - as represented in the initial data base - could be reconciled. Future evolution of the same technique will provide an interactive CAD-type creation of new object models and data bases, which can be further used for robot manipulation under telerobotic control.

The "Object Designation" process, of e.g. designating a box, is initiated by OCS requesting the object model of the box from TPR, where the data base of all concerned objects resides (or at least, TPR can access to). The object model basically contains a name list of all the vertices and edges of the model, a base frame of the model (e.g. of the centroid of the object), and the list of transformations of the vertices relative to the base frame. Along with the object model, the OCS obtains the camera model from TPR, including the focal length, optical axis pointing vector, and the transformations that convert world coordinates into camera image coordinates.

The OCS Object Designation software then computes the pixel coordinates of the box (i.e. the vertices and edges of the box) so as to overlay its wire frame properly in location and orientation onto the selected camera image. If the object does not fall in the present field of view of the camera, OCS software will conjure the object model on the camera image as presented.

The Operator now verifies that the object model, as represented in the data base, is or is not correctly placed, by examining the model's wire frame overlay relative to what is shown in the camera image corresponding to the same object. If the two do not coincide, the Operator will start an interactive mouse point-and-designate process to associate vertices in the wire frame to those of the real image. After successively associating three or more vertices, and repeating the same on more than one camera view of the object, the Operator can call a best-fit procedure. This will cause the computation of the object transformation, containing the actual object location and orientation (relative to the world). Upon acceptance by the Operator, this now updated position of the object will be sent to and stored in the telerobot data base. Telerobot task execution can then proceed using this updated data base.

Figure 7 shows the wire frame of a box, which is being overlayed and designated onto the image of the box, as seen by the camera. The top of the figure shows the menu options for various image operations.

The hardware that performs this process is the New Media Graphics Generators discussed in Section 3.2.5 and shown in Figure 6. The OCS SUN computer contains all the computing software.

## 4.0 FUTURE RESEARCH AND DEVELOPMENT

In its present state of completion, the OCS is shown in Figure 8. The completed system is now planned for integration and installation at JPL's Telerobot Demonstrator Testbed laboratory in Spring, 1989. After its successful integration with the rest of the Telerobot System in Summer, 1989, the OCS will serve as the focal point of the Telerobot Demonstrator System. Real hands-on operational flow analysis and workload analysis will be conducted, so as to evaluate the effectiveness of the OCS design, and of the integrated telerobot system.

More research and development items, improvements on point-designs, alterations of physical layout, addition of vocabulary, etc. will undoubtedly surface when more experience is gained from OCS and telerobot experiments. Other already foreseen technology development items include: interactive model/data base building; the use of CAD-type data base techniques for object trajectories planning and verification; faster and better algorithms in object designation process, including hidden line removal, incorporation of perspective cues etc.; smoother and more unified operator-machine interface; more powerful display, iconic representations and graphics. As more powerful computers become available, and as the understanding of a telerobot system matures, the state-of-the-art OCS technology will evolve.

## 5.0 Acknowledgements

This work was performed at the Jet Propulsion Laboratory, California Institute of Technology, under contract to the National Aeronautical and Space Administration. GE Aerospace (formerly RCA) / Advanced Technology Laboratory was subcontractor to JPL, responsible for the development of the Operator Control Station and part of the Manipulator Control Subsystem.

Figure 1. TELEROBOT OPERATOR CONTROL STATION  
FUNCTIONAL BLOCK DIAGRAM

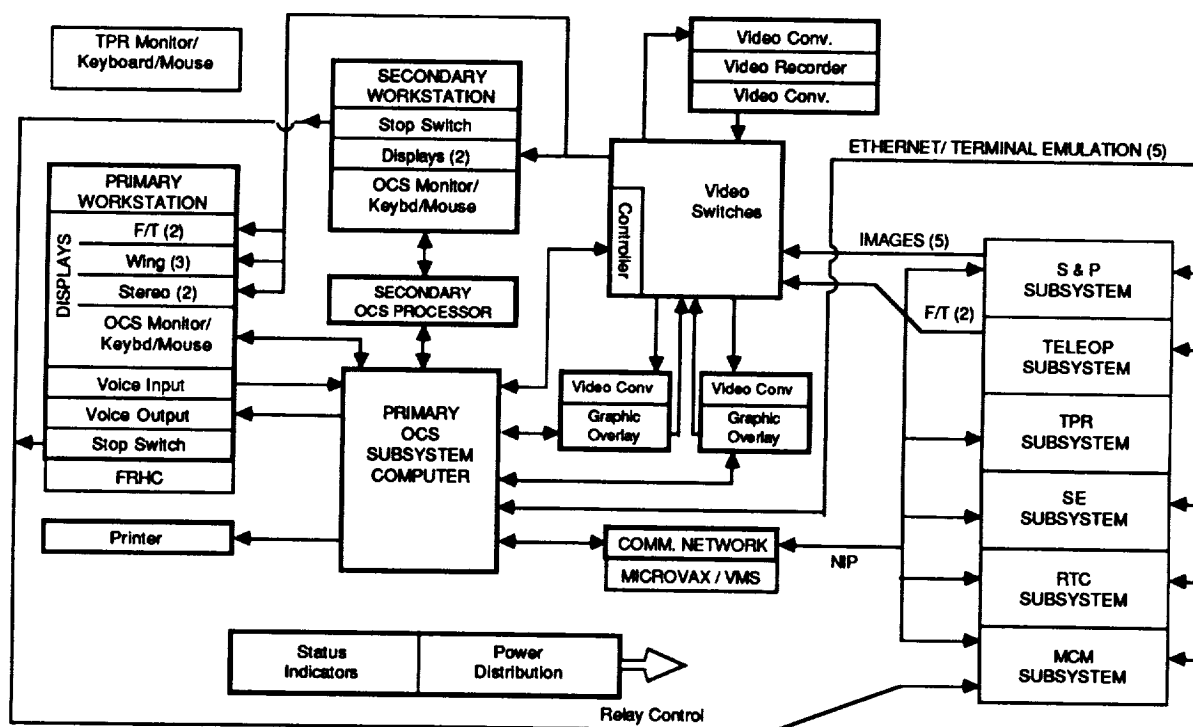
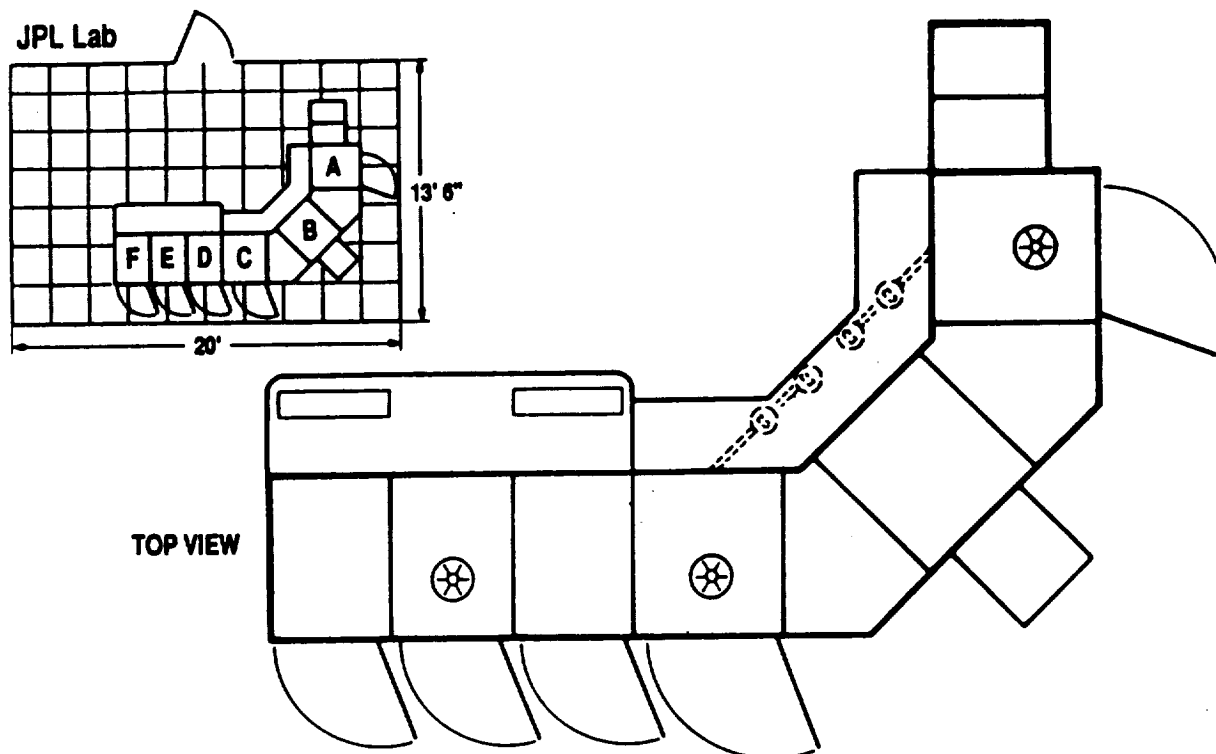


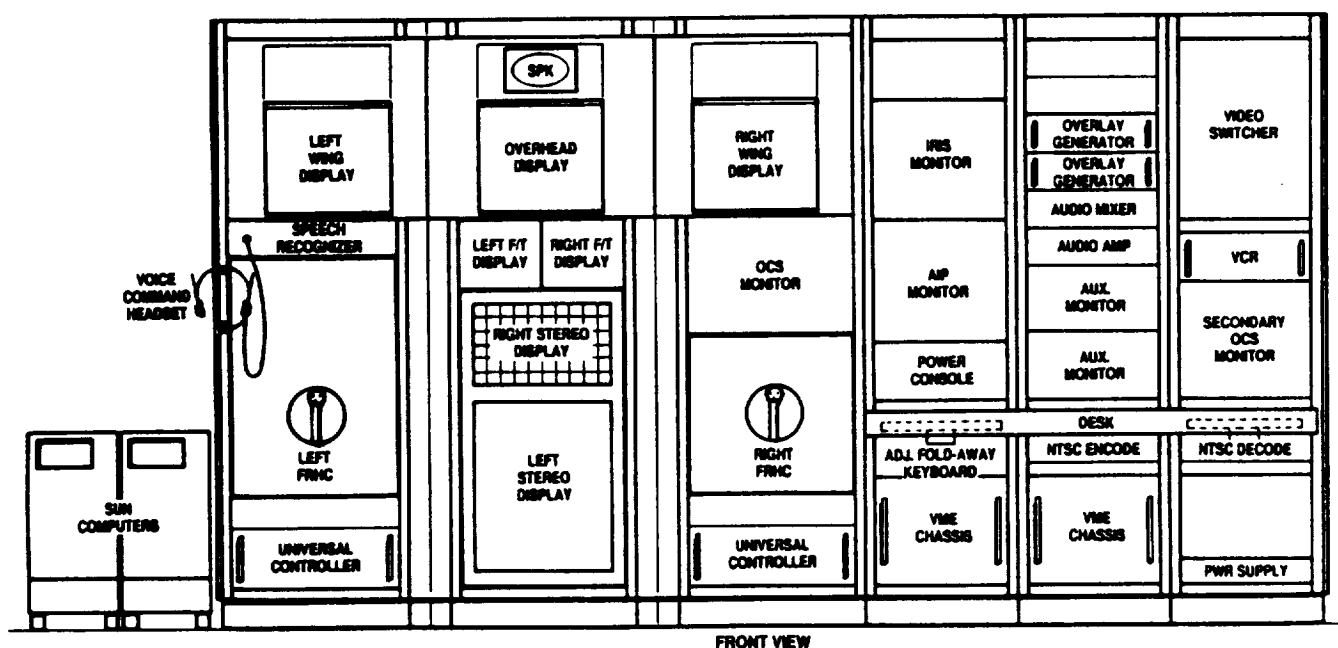


Figure 2. THE OCS - PLAN VIEW



TOP VIEW

Figure 3. THE OCS - ELEVATION VIEW



FRONT VIEW

Operator Control Station

ORIGINAL PAGE IS  
OF POOR QUALITY

Figure 4. HAND CONTROLLER RANGE OF ADJUSTMENTS

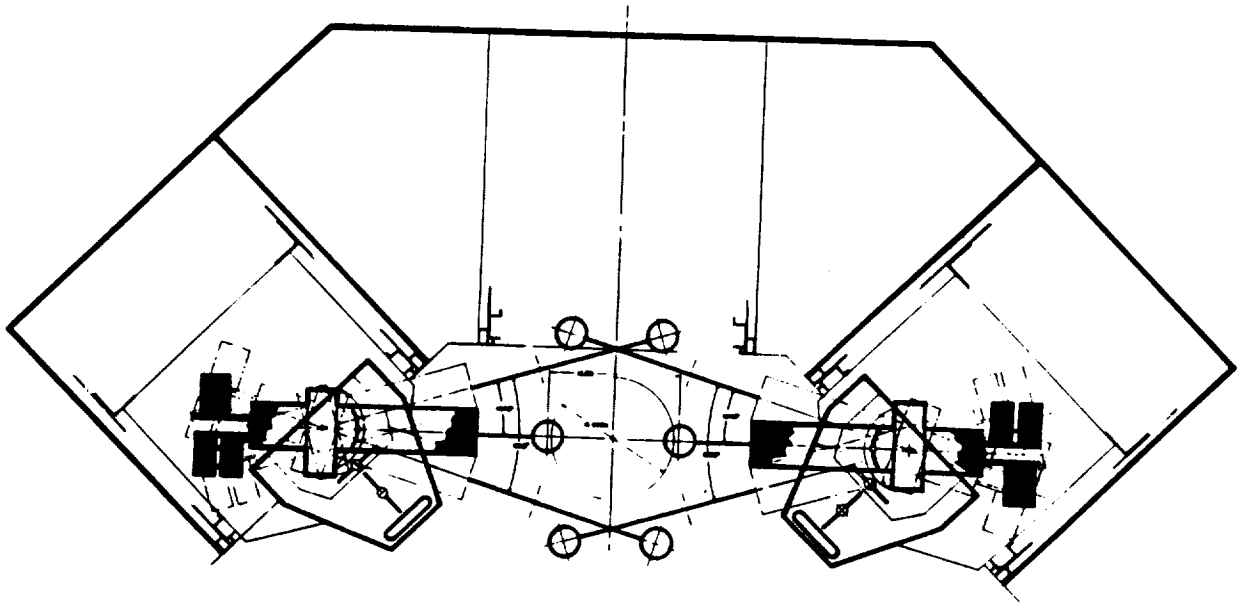


Figure 5. STEREO DISPLAY MONITOR ARRANGEMENT

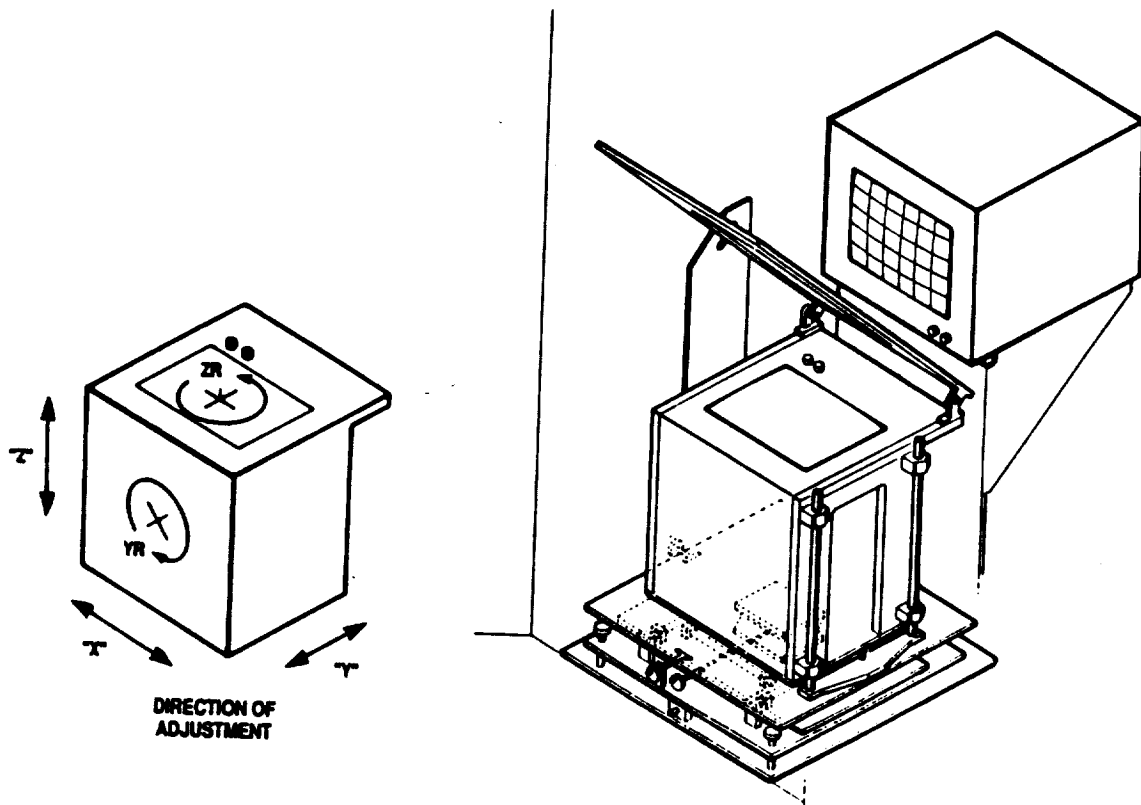


Figure 6. OCS VIDEO SUBSYSTEM ELEMENTS

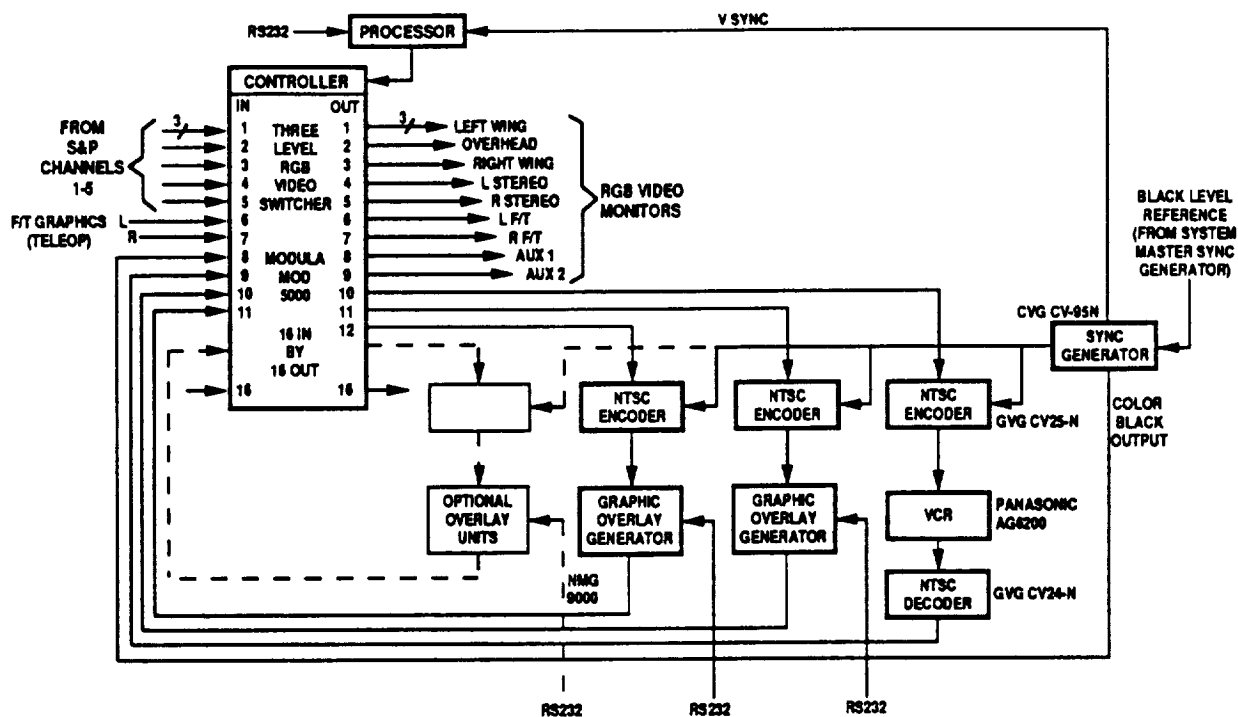
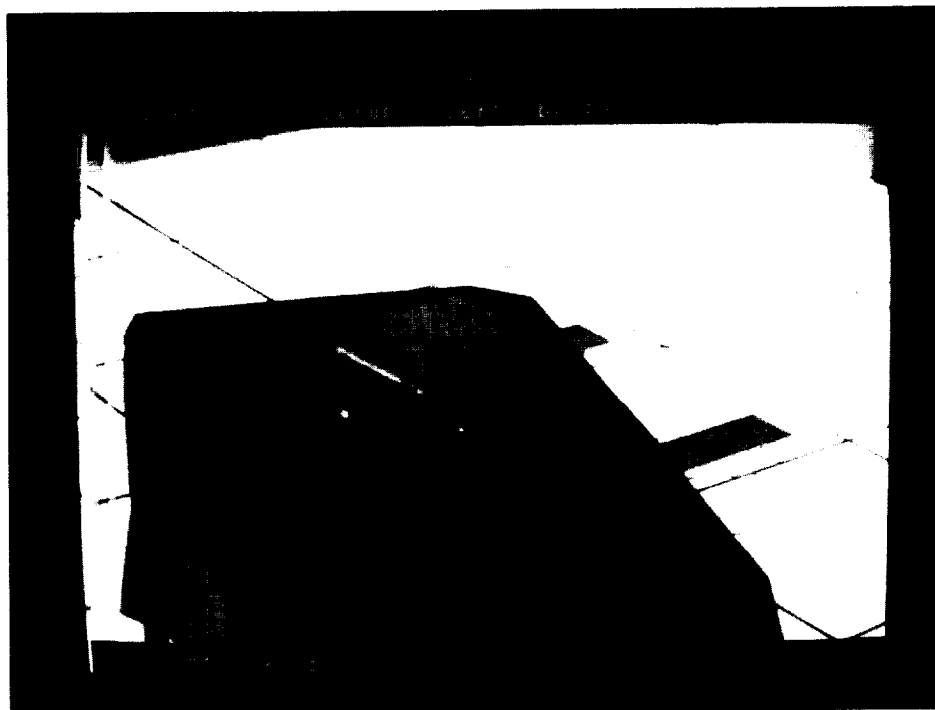


Figure 7. OBJECT DESIGNATION - WIRE FRAME OVERLAY



## Figure 8. THE OPERATOR CONTROL STATION

(Status as of 1/89. For complete layout of elements, refer to Figure 3.)



ORIGINAL PAGE  
BLACK AND WHITE PHOTOGRAPH

### 6.0 References

- [1] P. Schenker, "Telerobot Program Objectives and Technology Outreach," Proc. of the First Workshop on Space Telerobotics, Pasadena, Ca., Jan. 1987.
- [2] J. Oberright, "Space Station Flight Telerobotics Servicer Program Overview," Proc. of the First Workshop on Space Telerobotics, Pasadena, Ca., Jan. 1987.
- [3] E. P. Kan, "The JPL Telerobot Operator Control Station: Part II - Software," Proc. of the Second Workshop on Space Telerobotics, Pasadena, Ca., Jan. 1989.
- [4] R. Cain, et al., "Network Interface Package (NIP) User's Guide," SRI International, Contract #957908 Report to JPL, also SRI Project #3528, Nov. 1987.
- [5] J.N. Herndon, et al., "The state-of-the-art Model M-2 Maintenance System," Proc. ANS Topical Meeting on Robotics and Remote Handling in Hostile Environments, Gatlinburg, Tn, Apr. 1984, pp. 147-154.
- [6] D.P. Kuban and H.L. Martin, "An Advanced Remotely Maintainable Force Reflecting Servomanipulator Concept," Proc. ANS Topical Meeting on Robotics and Remote Handling in Hostile Environments, Gatlinburg, Tn, Apr. 1984.
- [7] M.M. Clark, W.R. Hamel and T.V. Draper, "Human Factors in Remote Control Engineering Development Activities," Proc. of 31st Conference on Remote System Technology, Detroit, Mi, 1983.
- [8] J. Matijevic, et al., "Functional Requirements for the Telerobotic Testbed," Jet Propulsion Laboratory, Document #JPL D-3693 (Internal Document), May, 1988.
- [9] E.P. Kan, "Telerobot Operator Control Station Requirements," Proc. of USAF/NASA SOAR '88 (Space Operations - Automation and Robotics) Workshop, Dayton, Oh., Jul. 1988.
- [10] S. Oxenberg, P. Landell, and E. Kan, "Geometric Data-Base Maintenance Using CCTV Cameras and Overlay Graphics," Proc. SPIE Symp. on Optical and Optoelectronic Engineering: Space Station Automation Conf., Cambridge, Mass., Nov. 1988.

## **The JPL Telerobot Operator Control Station: Part II - Software**

**Edwin P. Kan\***

**B. Patrick Landell^, Sheldon Oxenberg^, Carl Morimoto^^**

### **ABSTRACT**

The Operator Control Station of the JPL/NASA Telerobot Demonstrator System provides the man-machine interface between the operator and the System. It provides all the hardware and software for accepting human input for the direct and indirect (supervised) manipulation of the robot arms and tools for task execution. Hardware and software are also provided for the display and feedback of information and control data for the operator's consumption and interaction with the task being executed. This paper, Part II, addresses the software design of the OCS.

### **1.0 INTRODUCTION**

The JPL/NASA (Jet Propulsion Laboratory / National Aeronautics and Space Administration) Telerobot Demonstrator System is a research testbed for the development, integration and testing of advanced robot control technologies. The component technologies and system-wide design experiences derived from such a system development and technology demonstration are targeted for use in future space programs, including the NASA's Flight Telerobot Servicer project.

The Operator Control Station (OCS) is a part of this Telerobot Demonstrator System. It contains state-of-the-art hardware, both mechanical and computing, for providing control input to the System. It contains software, in controls as well as human operator interface, for real-time and user-friendly interaction. Video displays for text, graphics and camera images are provided for operator consumption; where appropriate, voice input/output is provided to reduce operator work-load. Data manipulation such as object designation capability is provided for efficient task definition and execution. Access to all Telerobot subsystems is provided for software development and on-line monitoring.

The OCS system design and hardware configuration have been discussed in details in Part I of this paper [ref.1]. This paper, Part II, concentrates on the software design.

\* Jet Propulsion Laboratory, California Institute of Technology, Pasadena, Calif.

^ GE Aerospace (formerly RCA) / Advanced Technology Laboratory, Moorestown, N.J.

^^ GE Aerospace / Western Systems, San Jose, Calif.

## 2.0 OCS SOFTWARE REQUIREMENTS

Software required in the OCS includes the processing of OCS input/output data; interface software with other subsystems (in the Telerobot Demonstrator System); system mode switching and supervision; and man-machine interface including object-data-manipulation. Other closely related controls software, including the processing of the force-reflecting-hand-controllers kinematics and dynamics, the processing of the force-torque displays, various other system management functions and supervisory control man-machine interface will not be discussed in here. This is because certain project partitions have allocated these functions in other subsystems in the overall system architecture design. Specific OCS software requirements are to provide:

1. Command interpreter to process operator generated commands via the keyboard, mouse, and voice; hence to parse, translate, and generate inter- and intra-OCS commands;
2. Message processor for translating, generating, and displaying messages on OCS monitor; for messages initiated from within or outside OCS;
3. RS-232 controllers for graphics overlay/mixer controllers, voice display controller, and video switch controller;
4. Gateway computer interface via an ethernet network;
5. NIP (a custom Network Interface Package [ref.2]) gateway interface software for processing NIP transactions from other subsystems;
6. Object designation/definition software to create wire-frame models, overlay on camera images, manipulate using mouse cursors, perform best-fit, update and augment data bases; and to interface with the Planning and Reasoning Subsystem for models and data;
7. Interface software between the primary workstation and secondary workstation;
8. Terminal emulation to all subsystems via multiple windows on the OCS monitor;
9. Pull-down and pull-right menus for system and subsystem commands;
10. Continuous speech voice recognition with error correcting and custom vocabulary grouping schemes for direct voice input; multi-voice (gender/person) speech synthesis for message output;
11. Graphics generation and overlay software for object designation/verification.

Other pertinent requirements on the OCS will not be discussed in this paper, because of their allocation to other subsystems in the present Telerobot Demonstrator System design. Nonetheless, these other requirements are enumerated in [ref.3].

Performance requirements on the OCS software include:

- i. Both the primary and secondary workstations shall have identical software and input-output process from the terminal, keyboard and mouse; (by design and because of hardware limitation, the secondary workstation cannot input FRHC teleoperation controls and does not have stereo vision displays);
- ii. Both the primary and secondary workstations shall share common software for executing and interfacing commands between the command interpreter/message display processes and the rest of the OCS internal/external software; this common software shall reside in the primary workstation;

- iii. Inputs from both workstations shall not be prioritized, and shall be processed sequentially by the time order that the inputs arrive;
- iv. Flexibility shall be maintained in the overall structure, grouping, and individual specification of the set of commands; the entire set shall be kept in a separate command definition file, which can be modified and edited independent of the compilation of the main OCS software;
- v. Similar flexibility shall be maintained in the set of messages; the entire set shall be kept in a separate message definition file, which can be edited independently;
- vi. The set of vocabulary words, grouping and grammar of the speech input sentences shall be designed to achieve 98%+ Type I recognition accuracy and 95%+ Type II false alarm rejection accuracy; with the use of interrogation and operator interaction, the overall accuracy achievable shall be improved to 99%+;
- vii. The accuracy of the object designation process shall be within one pixel RMS (root mean square) averaged over the vertices of the object; (absolute accuracy is not specified, because it is a factor of the distance of the object from the cameras and depends on camera model accuracies);
- viii. The primary OCS workstation mouse shall be shared between the normal OCS command process and the object designation process which utilizes the mouse for graphic entries; (this property eliminates the proliferation of mouse(s); each graphic overlay machine for the object designation process requires one mouse);
- ix. During the object designation process, OCS commands shall continue to be receivable through voice input; keyboard and mouse inputs can be toggled between the designation and command processes, using a switch on the mouse;
- x. Menus and direct keyboard inputs shall be designed to good human engineering standards.

### 3.0 OCS SOFTWARE DESIGN

The OCS software consists of ten processes distributed among a Sun-3/160, which serves as the primary workstation, a Sun-3/60, the secondary workstation, and a DEC  $\mu$ VAX, which is the communications gateway computer for interface among the other subsystems. These processes provide the following OCS functions:

- o Command Interpreter
- o Message Processing and Display
- o Ethernet Interface, Sun/microVAX
- o External Subsystem Interface
- o Video Switch Operation and Control
- o Wire Frame Object Designation

The dual workstation design allows OCS commands to be issued from either the primary or the secondary workstation, and provides for messages generated on either workstation, or by external subsystems, to be displayed on both workstation monitors. Voice input and audio output capabilities have been included to aid the operator. A continuous speech recognizer is provided to accept operator voice commands, which are routed to the OCS Command Interpreter. A speech synthesizer is provided to

acknowledge voice commands, as well as annunciate important messages that may occur when the operator's attention is directed away from the workstation display. Multiple voices are utilized to assist the operator in message recognition.

The processes which constitute the OCS program set are illustrated in Figure 1. The SunView Notifier is shown with the processes that provide direct window interface to the users by either accepting operator input or displaying messages. Inter-process communication methods used by the OCS software, include Remote Procedure Calls, Unix Sockets, Pipes, and DECnet. Figure 1 also shows the Ethernet dividing the primary from the secondary workstation, and separating the OCS software which resides on the Sun workstations from the  $\mu$ VAX gateway computer and external subsystems.

The OCS Command Interpreter resides on both the primary and the secondary workstation, and receives operator commands from the speech recognizer, as well as the standard Sun keyboard and mouse. Interpreted command data is sent to other OCS functions in the form of command interface records by means of Sun's Remote Procedure Call (RPC) and eXternal Data Representation (XDR) facilities, which provide for inter-process communication regardless of the host in which the serving process resides.

The OCS command interface, in addition to allowing direct keyboard and voice commands, provides a customizable multi-level menu interface that is controlled by the Sun three-button mouse. Menu display is activated by clicking a mouse button, or selecting an icon, and command selections are made by positioning the mouse cursor over the desired option. Many menu items contain a "pullright" capability, which, when selected by the mouse cursor, causes the next level of menu options to be displayed to the right of the current selection. The operator may specify, during OCS activation, the degree to which previous menu selections influence subsequent menu display.

The Message Processor accepts message definition records, which contain message ID's and message attributes, from all OCS processes, and generates messages for display on the Sun monitors and for annunciation by means of the DECtalk speech synthesizer. Messages targeted for monitor display are routed to the message display software on both the primary and the secondary workstations so that all messages are displayed on each workstation.

The Ethernet Interface function of the OCS provides the Sun interface with the  $\mu$ VAX gateway computer. This function accepts command interface records from processes on either Sun workstation, extracts pertinent data for interface with external subsystems, and sends host-to-gateway records over the Ethernet to the External Subsystem Interface function on the  $\mu$ VAX via DECnet. The Ethernet Interface function also accepts gateway-to-host records over the Ethernet from the External Subsystem Interface function, extracts pertinent data, and distributes the data to the appropriate OCS process by means of (1) command interface records using RPC facilities, (2) message definition records, also using RPC, and (3) data written on a Unix socket connected with the Video Switch Operation function.



The External Subsystem Interface function resides on the  $\mu$ VAX as the communication interface between the OCS software that executes on the Sun workstations and the external subsystems. This inter-subsystem communication is done using Network Service Transactions (NST's) which are controlled by the NIP. The External Subsystem Interface function receives host-to-gateway records over the Ethernet from the primary Sun host via DECnet services, and uses the data to initiate or respond to NST's with other TDS subsystems. In addition, NST records that originate from external subsystems are received and validated by this function. Validated data is extracted from the NIP NST's, formatted into gateway-to-host records, and sent via DECnet to the Ethernet Interface function on the Sun primary workstation.

The Video Switch Operation and Control function resides on the primary Sun workstation and provides the software interface to the OCS video switch hardware. The Video Switch Operation function interfaces with the external S&P Subsystem, by means of the OCS Ethernet Interface and External Subsystem Interface functions, to request remote switching of the cameras and frame buffers to operator selected video channels, and to receive the remote video switch status.

The Wire Frame Object Designator function resides on the primary Sun workstation and provides the capability for objects whose position and orientation are unknown or invalid to be designated and subsequently used in the System database. The designation is performed by means of wire frame diagrams, based on object and camera model data, being overlayed on actual video images, manipulated in terms of translation and rotation, and fit with the video image by means of vertex designation and a least-squares fit algorithm. The Wire Frame Object Designator function receives object and camera model data from the external TPR Subsystem, by means of the OCS Ethernet Interface and External Subsystem Interface functions, and returns updated camera model data after an object has been designated. In addition, object designation commands from the operator are received in command interface records sent by the OCS Command Interpreter process.

An extensive user interface for the Teleoperation Subsystem is provided to facilitate both operator input and status recognition for the robot manipulators and the force-reflecting hand controllers, their states, joint limiting situations, operating modes, etc. Numerous icons are provided for the selection of the various teleoperation command and status windows. A direct communication link with the Teleoperation computer is provided, and voice teleoperation commands received by the Command Interpreter function are routed to the Teleoperation Subsystem by this link.

The OCS software has been designed to operate within the SunView Windows user interface environment. The OCS window layout can be customized by the operator to a large degree, including the size and position of the windows, the default character font, icon positioning, the look of menus, etc. The windows may be moved about the Sun display, resized, or closed into icons, and many window option defaults may be selected according to the operator's desires. Easily identifiable iconic representations for each of the OCS windows have been provided, and the mouse cursor image is modified to indicate the active window in order to give additional visual cues to the operator.

Integrating the OCS operator interface with the SunView environment also provides a flexible capability for the operator to perform terminal emulation with all the external subsystems. Terminal emulation windows may be opened when direct interface with another subsystem is required, and subsequently destroyed, or left active in an iconic state for future use. Multiple terminal emulation windows may be active simultaneously, in addition to the OCS processes.

This implementation approach allows the operator a great deal of flexibility in the user-interface with the OCS, and provides a familiar look and feel for a user experienced with the Sun window environment.

The following three subsections further describe the Command Interpreter process, the Message Definition and Display process, the Ethernet and External Subsystem Interface process. Explicit discussion on the Video Switch Operation and Control is avoided because of the simplicity of the process. Discussion on the Object Designation process has been given in [ref.1,4].

### **3.1 Command Interpreter**

Table 1 is a condensed table of the OCS command set. The OCS Command Interpreter function provides a flexible, table driven capability for command definition, recognition, interpretation, and inter-process distribution. Valid commands for direct keyboard entry, voice input, and menu selections, as well as command and command-qualifier relationships, are defined in a standard text file, the command definition file, and can be modified by the user to change the command interpretation and processing traits of the OCS without modification of the software.

The command definition file completely defines the command processing characteristics of the OCS. Specific keyboard and voice commands are defined, as are command groupings, menu content, and the text of menu selections. Command translation data is defined for process specific or device specific information to be supplied in conjunction with a particular command. The command destination--that is, the target program that will process the command, is also specified for each command, as is any required NST data associated with the command, such as the destination subsystem, record type and identification, and subsystem interchange protocol. Each command may be specified as requiring confirmation before being processed, providing a measure of protection from inadvertent command selection.

The purpose of the command definition file and the Command Interpreter flexibility is to allow the OCS and Telerobot Demonstrator System functionality to evolve, without the necessity of redesigning or modifying the command processing function. Additional processes which may be added to the OCS at a later date, will be able to use the same user interface, yielding consistency as the System capabilities evolve. Commands may be grouped into menus, and the text of the menus may be modified after the user has gained experience with the system and becomes aware of changing needs. In fact, multiple command definition files may be defined to allow

each operator to customize his or her interface with the OCS according to individual preference or the tasks to be performed.

The operator interface to the Command Interpreter function makes use of the SunView Window services, and the SunView Notifier, which directs processing control based on user input by direct keyboard entry, mouse-based menu selections, or voice input. Voice input from the speech recognizer is directed to the Command Interpreter function, regardless of which SunView window is active.

Once the Command Interpreter has received a command from the operator, the command is looked up in the command definition table, which is built at initialization from the command definition file. Commands and their associated qualifiers are validated, and command confirmation is requested when so indicated. Command validation or cancellation may also be performed by use of keyboard or voice input, or by clicking the mouse buttons. For valid, confirmed commands, the data in the command definition table is used to build a command interface record, which is then sent to the specified destination for processing.

The Sun RPC (Remote Procedure Call) services are used for inter-process communication between the Command Interpreter, which resides on both the primary and the secondary workstation, and the other OCS functions, which operate on the primary workstation. Together with the XDR (eXternal Data Representation) services provided by Sun, which organize data bytes in a machine independent format, RPC allows an inter-process communication client process to communicate with a server process on any valid host computer. These facilities enable the Command Interpreter to execute on either workstation without requiring special-purpose software for network interface from the secondary workstation's Command Interpreter to OCS functions that reside on the primary workstation.

### **3.2 Message Processing and Display**

Table 2 is a condensed table of the OCS message set. The OCS Message Processing and Display function provides a centralized, table driven interface to the OCS message windows on the Sun workstations, and the voice output device.

The Message Processing server process for RPC from other OCS functions resides on the primary workstation as the target of all Message Definition Records. This process builds the message text based on the incoming data and the specification in the Message Definition Table, and routes the message text to the DECtalk speech synthesizer, and the Message Display software residing on both the primary and the secondary workstations. Audio output may be enabled or disabled by normal OCS commands, and the gender of the message voice is selected based on the attributes specified in the Message Definition Table.

A standard ASCII format Message Definition File is used for the specification of message ID's, attributes, and text, which are processed into the Message Definition Table during initialization. The message attribute specification allows a message to be defined as a normal or critical text message, and/or a male voice, or female voice

message. The use of the Message Definition File allows messages to be added or changed, or the characteristics of a message to be revised based on evolving system needs, without requiring redesign or modification of the existing OCS software.

A separate message window is utilized to inform the operator of critical messages. The display of a critical message is accompanied by an audible beep, a visual flashing of the critical message window, and if the message window had been closed into an icon or hidden behind another window, it is automatically opened and exposed to operator view. Both critical and normal messages are displayed in the normal message window, which provides a context for any critical messages that may be generated during an operational session.

All valid commands are acknowledged by messages in the normal message window, and all messages are displayed at both the primary and the secondary workstation with an indication of the workstation that generated the message. These features allow each OCS workstation operator to be aware of the actions of the other operator, as well as any system or error messages that arise, and provides a history of commands and how they were generated.

The capability to store OCS messages from both the critical and normal message windows to a user specified file is provided, along with the option for the operator to clear each message window individually - to eliminate outdated critical messages, for example. The workstation mouse is used to activate a menu and select the desired option.

The OCS message windows may be closed in icons or positioned on the workstation display independently from the other OCS windows, such as the command window. This flexibility allows the OCS message display to be rearranged according to individual operator preferences or processing needs.

### **3.3 Ethernet and External Subsystem Interface**

The Ethernet Interface and External Subsystem Interface functions of the OCS work together to provide a centralized interface between the Sun workstations and the  $\mu$ VAX gateway computer, and between OCS and the external subsystems.

The Ethernet Interface function consists of two separate processes, one for sending host-to-gateway records from the Sun to the  $\mu$ VAX, and the other for receiving gateway-to-host records from the  $\mu$ VAX. Upon initialization of the Ethernet Interface function on the Sun primary workstation, DECnet services are used to activate the External Subsystem Interface function, which executes on the communications gateway  $\mu$ VAX, and establish a communication link over the Ethernet between the two functions.

The External Subsystem Interface function receives host-to-gateway records from the Sun, extracts the data and builds the NIP/NST records to be sent to the other subsystems, then interfaces with the NIP to ship the NST records. All interface with the NIP software is performed in this function, which also receives NIP/NST records

directed to the OCS from the other subsystems. Data is extracted from the incoming NST records, packaged into gateway-to-host records, and sent to the Ethernet Interface function over the Ethernet.

The Ethernet Interface function receives the gateway-to-host records sent from the External Subsystem Interface function, and examines the record content to determine which OCS process should receive the data. Message data is sent by means of RPC's to the Message Processing function. Other data is used to generate Command Interface Records, which are also sent via RPC to the destination process. Video switch status from the S&P subsystem is sent over a connected Unix socket to the Video Switch Operations and Control function, which is waiting for the status in order to complete a video switch setting operation, while object and camera model data sent by the TPR subsystem is distributed to the Wire Frame Object Designation function using an RPC interface.

Current design contains the following NST's. (OCS-to-SE): (i) Comm\_Link\_Test; (ii) SE\_Start-Up, (iii) SE\_Shut\_Down, (iv) SE\_Halt. (OCS-to-S&P): (v) Select\_Video\_Sources, (vi) Read\_Video\_Sources. (OCS-to-TPR): (vii) Vision\_Arm\_Control, (viii) Object\_Designate. (OCS-to-all): (ix) Message.

## **4.0 SUMMARY**

As mentioned in Part I of this paper, the present OCS design is an evolutionary design which will evolve and change as the telerobot technology matures, both in system design and in component design. The present design is believed to have the necessary 'hooks and scars' for future system expansion, both in software and in hardware. Despite its flexibility, certain architectural features are recognized to be suboptimal because of project constraints. These constraints have been discussed in Part I of this paper.

After complete integration in the Telerobot Demonstrator System testbed laboratory in Summer, 1989, the OCS will serve as the focal point of the Demonstrator System. Real hands-on operational flow analysis and workload analysis will be conducted, so as to evaluate the effectiveness of the OCS design, and of the integrated telerobot system. More research and development items, improvements on point-designs, alterations of physical layout, addition of vocabulary, etc. will undoubtedly surface when more experience is gained from OCS and telerobot experiments. As more powerful computers become available, and as the understanding of a telerobot system matures, the state-of-the-art OCS technology will evolve.

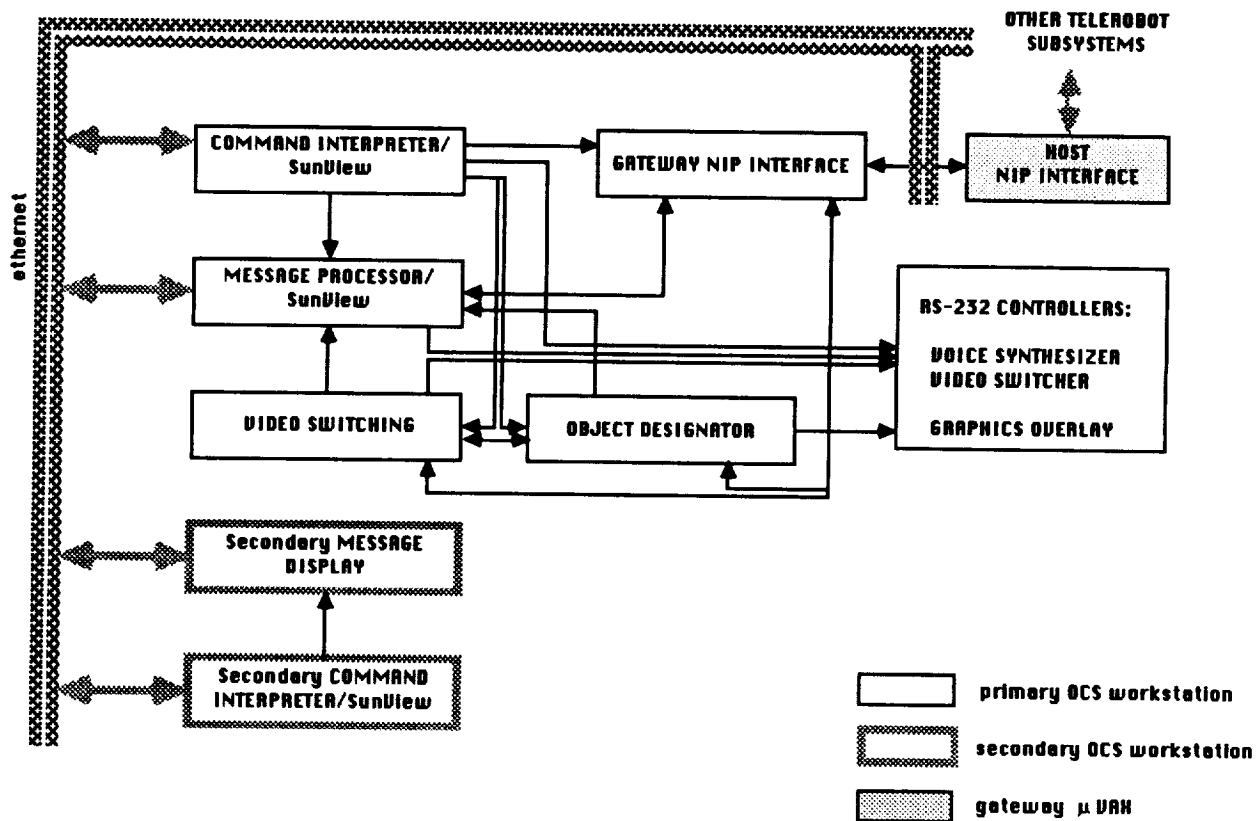
## 5.0 Acknowledgements

This work was performed at the Jet Propulsion Laboratory, California Institute of Technology, under contract to the National Aeronautics and Space Administration. GE Aerospace/ Advanced Technology Laboratory and Western Systems were subcontractors to JPL, responsible for the development of the Operator Control Station and part of the Manipulator Control Subsystem.

## 6.0 References

- [1] E.P. Kan, J. Tower, G. Hunka and G. VanSant, "The JPL Telerobot Operator Control Station: Part I - Hardware," Proc. of the NASA Conference on Space Telerobotics, JPL Publication 89-7 (this proceedings), Pasadena, Ca., Jan. 1989.
- [2] R. Cain, et al., "Network Interface Package (NIP) User's Guide," SRI International, Contract #957908 Report to JPL, also SRI Project #3528, Nov. 1987.
- [3] E.P. Kan, "Telerobot Operator Control Station Requirements," Proc. of USAF/NASA SOAR '88 (Space Operations - Automation and Robotics) Workshop, Dayton, Oh., Jul. 1988.
- [4] S. Oxenberg, B. Landell, and E. Kan, "Geometric Data-Base Maintenance Using CCTV Cameras and Overlay Graphics," Proc. SPIE Symp. on Optical and Optoelectronic Engineering: Space Station Automation Conf., Cambridge, Mass., Nov. 1988.

Figure 1. OCS SOFTWARE FUNCTIONAL BLOCK DIAGRAM



**Table 1. OCS Command Set (condensed) 'KEYBOARD' & "Voice"**

<u>HOST CONTROL:</u>	'Quit OCS' ("Quit OCS"); 'Audio' ("Enable Audio"); 'Noaudio'; 'Audio Ack'; 'Noaudio Ack'; 'Vsdflt'; 'Swst'
<u>VIDEO SWITCH CONTROL:</u>	'C1' ("CameraOne"); 'C2'; ... ; 'C5' 'B1' ("BufferOne"); 'B2'; ... ; 'B4' 'CH1' ("ChannelOne"); 'CH2'; ... ; 'CH5' 'RFT' ("RightForceTorque"); 'LFT' 'STE' ("StereoCameras"); 'OH' ("OverheadCameras") 'RW' ("RightWing"); 'LW' 'WF' ("WireFrame"); 'DROP' ("Drop") 'SWST' 'parameters' ("SwitchStatus" "parameters")
<u>OBJECT DESIGNATION:</u>	'conj'; 'top'; 'bot'; 'left'; 'right'; 'rear'; 'rotate'; 'manip'; 'fit'; 'undo'; 'erase'; 'done'; 'cancel'; 'next'; 'color'; 'wf_cont'; 'select'; 'clear'
<u>VISION ARM CONTROL:</u>	'va'; 'vastop'; 'hold'; 'a' ("moveagain"); 'back'; 'goback'; 'wm' ("WorldMode") 'tm' ("ToolMode"); 'P1' ("Movetoposition1") 'P2'; ... ; 'f'; 'b'; 'r'; 'l'; 'u'; 'd'; 'pr'; 'pl'; 'tu'; 'td'; 'mm' ("move- more"); 'sa'; 'lm'; 'ls'; 'bm'; 'ss';
<u>TELEOP CONTROL:</u>	'teleop'; 'telestop'; 'idx'; 'lidx'; 'ridx'; 'pos'; 'lpos'; 'rpos'; 'rat'; 'lrat'; 'rrat'; 'jnt'; 'ljnt'; 'rjnt'; 'crnt'; 'lcrnt'; 'rcrnt'; 'tst'; 'ltst'; 'rtst'; 'one'; 'two'; ...
<u>SYSTEM EXEC COMMANDS:</u>	'systart'; 'shutdown'; 'halt';

**Table 2. OCS Message Set (condensed) 'Message window' & "Voice"**

MESSAGE DEFINITION FILE LISTS ALL MESSAGES BY: MESSAGE ID; ATTRIBUTE; AND TEXT.  
ATTRIBUTES ARE: (1) normal; (2) critical; (4) male voice; (8) female voice.

<u>INTERNAL MESSAGES:</u>	'quitting'; 'initializing'; 'input error on %s'; 'command %s is not recognized' ("%s is not recognized"); 'error - getting OCS video switch routing status %s'; ...
<u>VIDEO SWITCH warning:</u>	'unknown video switch name %s' ("named video switch not known"); 'invalid video switch command'; ...
<u>WIREFRAME OB DES errors:</u>	'bad camera # received' ("requested selection not done"); ...
<u>WIREFRAME OB DES warnings:</u>	'unknown obj des command %s' ("illegal command"); 'cannot fit, no points saved' ("no points saved"); .....
<u>WIREFRAME OB DES status:</u>	'object designator active' ("object designator active"); 'mouse being used for object designation'; .....
<u>VDT MONITOR warnings:</u>	"no point to erase!"; "showing different camera view!"; ....
<u>VDT MONITOR status:</u>	"%s points saved!"; "object rotated to show top view"; "undo done!"; "object partially out of camera view"; ...
<u>SUBSYSTEM messages:</u>	'MCM requires attention' ("the MCM requires attention"); ...; 'va limit stop joint 1'; ...; 'SE status: warning (1)'; .....





# DESIGN OF A MONITOR AND SIMULATION TERMINAL (MASTER) FOR SPACE STATION TELEROBOTICS AND TELESCEINCE

L. Lopez, C. Konkel, P. Harmon,\* S. King

Teledyne Brown Engineering  
300 Sparkman Drive  
Huntsville, Alabama 35807-7007

\*System Dynamics, Inc.  
200 Sparkman Drive  
Huntsville, Alabama 35805

## Abstract

Based on Space Station and planetary spacecraft communication time delays and bandwidth limitations, it will be necessary to develop an intelligent, general purpose ground monitor terminal capable of sophisticated data display and control of on-orbit facilities and remote spacecraft. The basic elements that make up a Monitor and Simulation Terminal (MASTER) include computer overlay video, data compression, forward simulation, mission resource optimization and high level robotic control. Hardware and software elements of a MASTER are being assembled for testbed use.

Applications of Neural Networks (NNs) to some key functions of a MASTER are also discussed. These functions are overlay graphics adjustment, object correlation and kinematic-dynamic characterization of the manipulator.

## 1.0 Introduction

Teledyne Brown Engineering (TBE) is completing the independent research and development of a Monitor and Simulation Terminal (MASTER) for use as a Space Station telescience terminal. This effort complements the TBE responsibility for outfitting the Space Station U.S. Laboratory. The terminal is being used within the Robotics Laboratory to investigate techniques to enhance space experiment platform resources such as:

- 1) Communications bandwidth
- 2) Crewtime
- 3) Microgravity

The MASTER is a key element in this effort because it provides the ability to conduct telescience prior to flight and also execute later remote operation of actual equipment and robot systems. The functions that will typically reside in the MASTER are:

- 1) Video Processing and Overlay
- 2) Predictive Simulation
- 3) Intuitive User Interfaces
- 4) Decision Assistance
- 5) 3D Modeling and Graphics

The goal of this effort is to provide a telescience platform which can be tailored to specific experimenter resource and operational requirements. The results of this activity will allow a better understanding of user requirements and improve the design of teleoperated experiments.

## 2.0 Monitor and Simulation Requirements

Predictive Simulation requirements for a MASTER system were first alluded to by Ferrell in 1965 [1]. More recently, Akin [2] and Konkel [3] et al have discussed the importance of predictive visual and force reflective simulation. The simulation and 3D modeling features of the testbed are based on these requirements. Initially our effort has been limited to a predictive visual simulation due to the complexities of predictive force reflection. Our design is also driven by user requirements such as:

- 1) Maximizing the amount of useful information while minimizing on-screen "clutter"
- 2) Minimizing head motion to different displays.
- 3) Providing a mode for uplink control of the manipulator during special situations or contingencies.

## 3.0 System Design

Basic design requirements for a MASTER are shown in Figure 1. Primary elements include the Simulation Platform and Monitor. The Simulation Platform communicates with the

remote site, which includes both robotic and automation elements, as shown in the upper portion of Figure 1.

A simulation-based testbed was chosen for validating algorithms and state of the art control concepts. In this manner, new teleoperations and remote monitoring methods can be easily integrated and adjusted. The task of integrating graphics terminals, video cameras, neurocomputers, and robot manipulators, however, is non-trivial. Although each hardware component is capable of individually meeting its own functional requirements, the combined system must be properly integrated.

The TBE Robotics Laboratory provides a simulated remote work site. This will allow MASTER to ultimately become part of an actual integrated telepresence testbed. This includes hardware in-the-loop to simulate time-delayed teleoperations.

A great deal of attention has therefore gone into the design of a "friendly" interface. The operator can utilize a range of input devices. Mouse, keyboard, voice and miniature master manipulators have been used as control inputs. To operate effectively in the presence of time delay, all user inputs can be processed by the predictive simulation. The path of the robot can be modeled by the simulation, in real time, based on the user's inputs. An attempt has been made to minimize the number of displays required. Ideally, a single high-resolution terminal with multiple (possibly voice controlled) overlay windows would suffice. Dual orthogonal camera views using windows were critical during robot control. MASTER also provides an optimum human-machine interface such that an untrained tele-experimenter can become a competent tele-operator within a short time.

The functional design of the MASTER Simulation Platform is shown in the block diagram of Figure 2. In order to have a useful predictive simulation an accurate dynamic and kinematic model of the remote manipulator is provided. In addition, the 3-dimensional perspective of the simulation is automatically aligned with live video from the remote location. The graphics engine, world model, math models (i.e., kinematic/dynamic manipulator models), and neural network models are all used by the predictive simulation.

A high-resolution graphics workstation, with its own geometric pipeline processor, forms the heart of the Simulation Platform. The graphics engine, math models and world models all reside within this computer. Video overlay is accomplished by simulating both the "live" and the predictive views or by the use of live video data and a frame grabber-mixer. The

neural network models reside on a PC based neurocomputer workstation. The neurocomputer hosts a scientific vector processing board and associated neural network simulation software.

#### 4.0 Work in Progress

This section briefly discusses our current work in progress. We are emphasizing neural computing applications for use in the MASTER.

#### 4.1 Neurocomputing

Neurocomputing methods and hardware have been applied to the predictive display system. Two such problems are perspective adjustment for accurate simulation overlay and adaptive modeling of the manipulator dynamics. Other areas under consideration are two-dimensional and three-dimensional image tracking and compression, limited resource allocation and force-eye-hand coordination.

We have demonstrated an overlay perspective calibration technique that uses NNs. The problem is to align the perspective of two superposed images so that they appear to be the same image. Figure 3 schematically illustrates the problem we are addressing and the neural controller. Given visual data from an on-orbit (remote) site and a simulated view of that site, control of the simulated perspective until the images overlap is required. The method demonstrated uses a vision computer to digitize each image separately. These images are then processed to provide a measure of translational and scale offsets between the images. These offsets along with the control commands that an operator would use to manually align the images are then given to a NN that uses the generalized delta learning rule of Rumelhart [4] (known as back propagation) to "learn" how to align the images as an operator would. We have been successful in training our system to do perspective line-up to within 5% overlap. A simple logic algorithm has also been developed that will perform this same task. On comparing this algorithm to the neural architectures, it was found that the logic algorithm can be considered a computational subset of the neural network. Figure 4 illustrates the three perspective calibration techniques investigated.

Dynamics modeling is possibly the most difficult problem faced by a predictive simulation. The low gravitational environment in space telerobotics may significantly alter the dynamic response of manipulators. If the manipulator dynamic response is not properly compensated for in the predictive model the simulation may lose its effectiveness. TBE has

begun experimenting with NN modeling of the kinematics and dynamics of two-dimensional manipulators. Application of Kohonen's neural model for vector quantization and self-organization [5] has yielded encouraging results. Figure 5 shows how a Kohonen NN can represent the kinematic workspace. The web in Figure 5 is generated by plotting the actual synaptic coupling weights (two per neuron) as two-dimensional points. Lines are then drawn to connect topologically nearest-neighbor neurons. Initially the weights are random; thus the net appears tangled. During learning, random joint angles are used to reposition the arm for each cycle. As learning proceeds the network weights form a representation of the kinematic workspace. Once the neural workspace is created, end effector position vectors along a trajectory can be used to stimulate the trained neurons. This will result in unique time-varying neural activation patterns for every possible trajectory. Figure 6 gives a qualitative view of a typical activity pattern over time. The amount of nodal stimulation will depend on many factors such as trajectory path, speed, acceleration, and neural time constants. We feel that this type of kinematic/dynamic modeling may be useful in the analysis of dynamic changes that can occur under different working environments.

To date, other internal research at TBE has developed a method for constructing practical optical processing components that will eventually provide a MASTER with very high-speed, cost-effective neural network computing power. These include a fixed interconnect optical neural network, a small rugged optical correlation device, and optical associative memory.

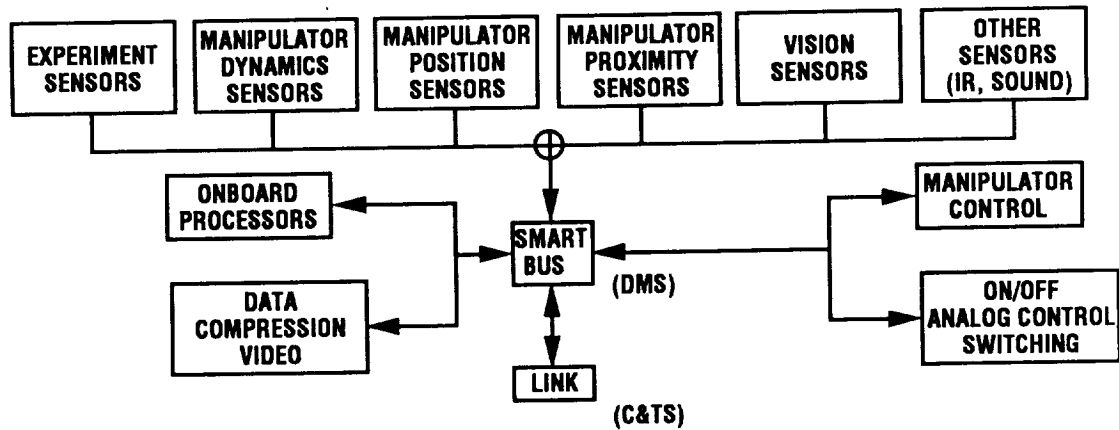
## 5.0 Summary and Conclusions

Efficient man-machine telescience is the goal of this effort. In order to meet such a goal the problem must be viewed from the end user's point of view. We have concluded that a good man-machine interface for practical telescience will require a sophisticated predictive simulation and monitoring platform that is transparent to the user and second nature to operate. This will require the application of new machine intelligence technologies, particularly neural computing.

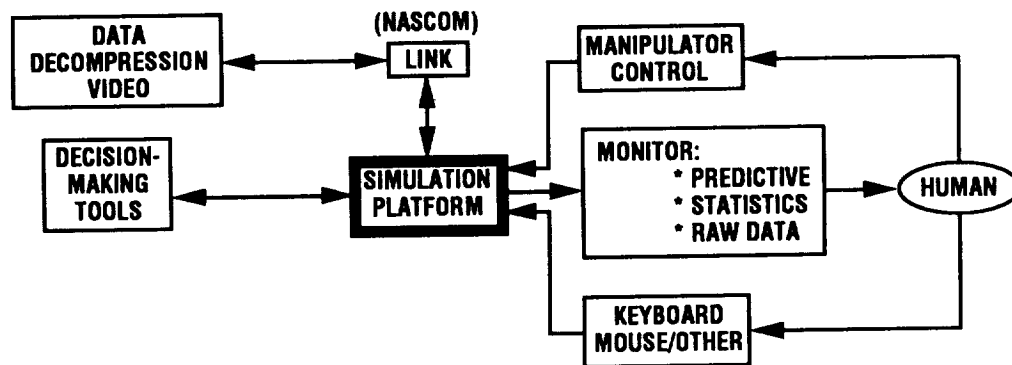
## 6.0 References

- [1] Ferrell, W. R., "Remote manipulation with transmission delay," IEEE Trans. Human Factors in Electronics, vol. HFE-6, pp. 24-32, Sept. 1965
- [2] Akin, Howard and Oliveira, "Human factors in space telepresence," NASAContract NASW3797, SSL#41-83, October 1983
- [3] Konkel and Miller, "Telerobotics and Orbital Laboratories: An end-to-end analysis and demonstration," IAF-87-27, IAF Congress, October 1987
- [4] Rumelhart, McClelland and the POP group, Parallel Distributed Processing, MIT press, Cambridge, Mass. (1987)
- [5] Kohonen, "Self-organization of topologically correct feature maps," Biological Cybernetics, 43, 59-69 (1982)

# REMOTE SITE:



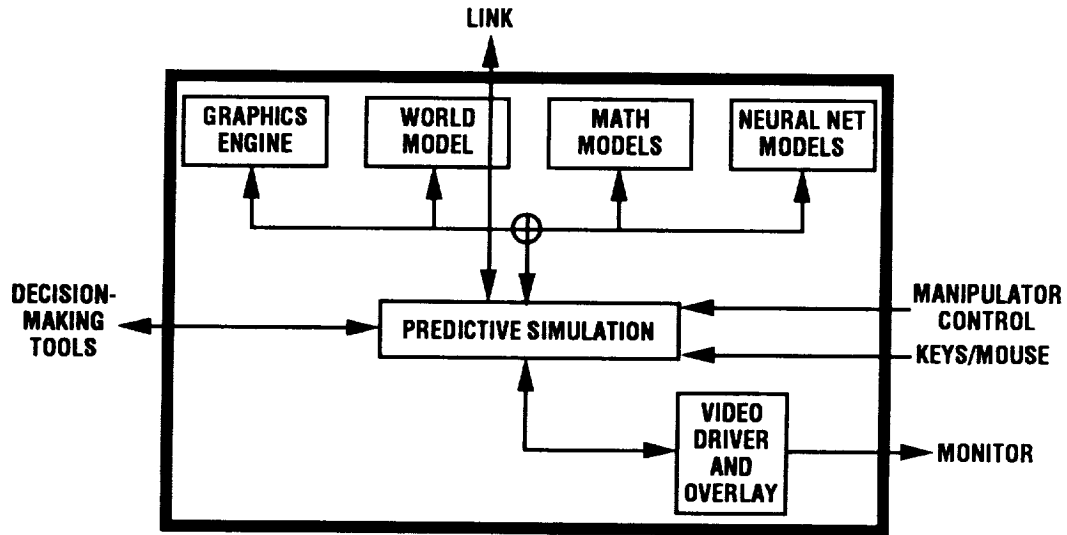
# GROUND: MASTER



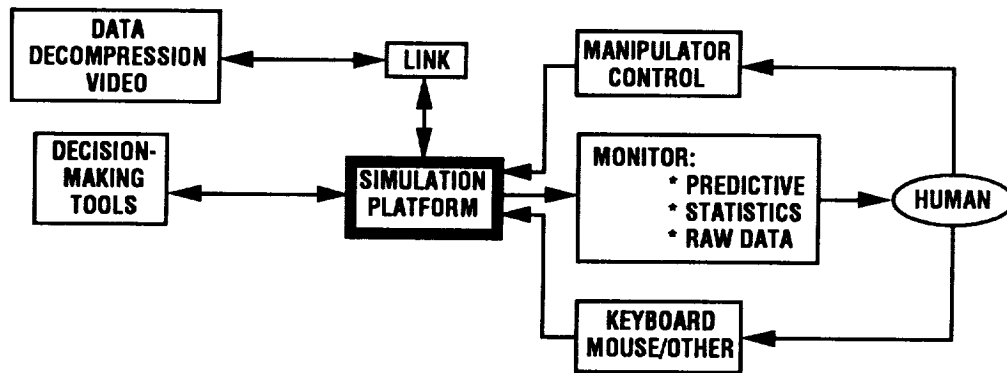
890207-002M3-11

FIGURE 1. BASIC REQUIREMENTS FOR A MASTER SYSTEM AND REMOTE WORK SITE

## INTERNAL VIEW: SIMULATION PLATFORM



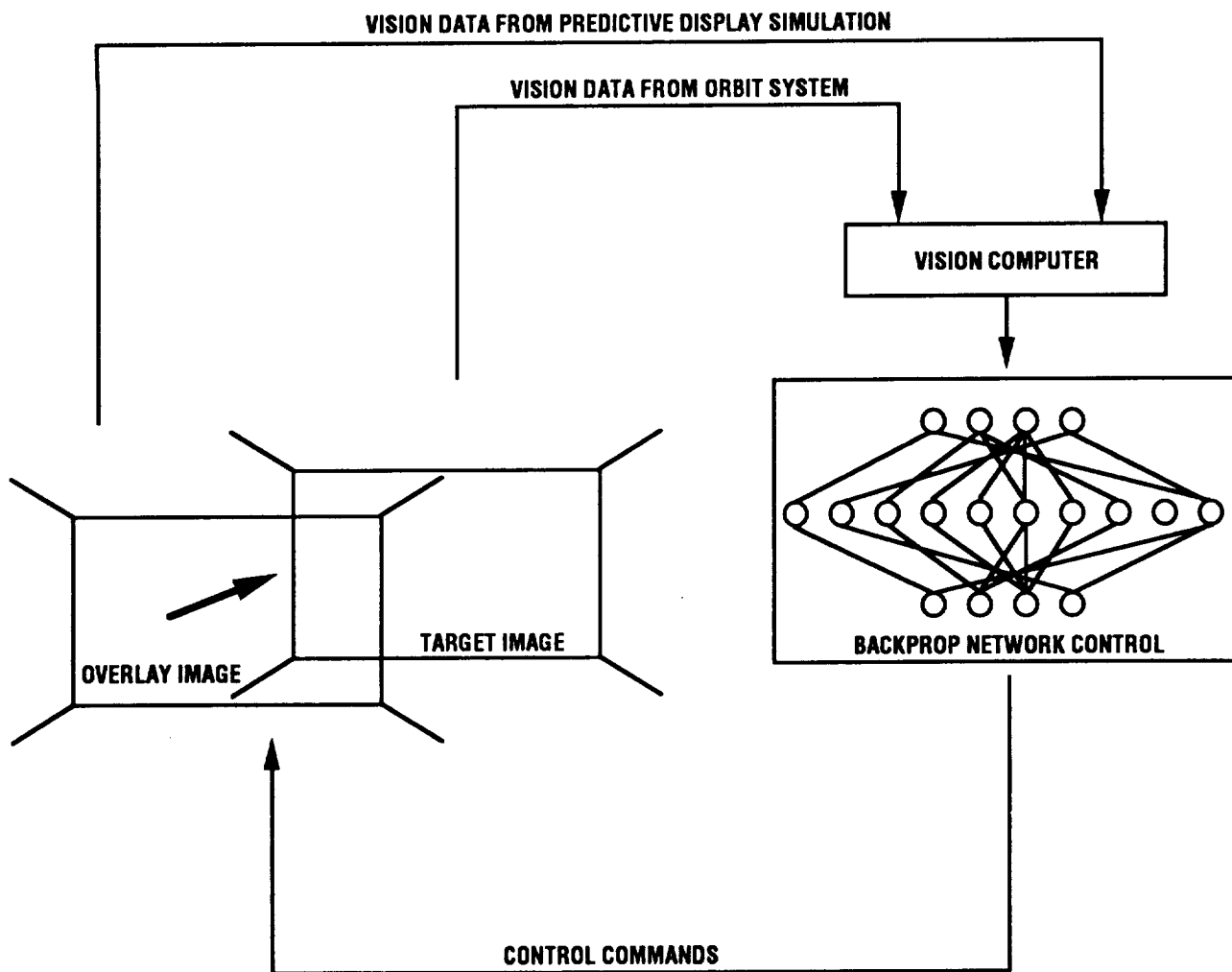
## EXTERNAL VIEW: MASTER



890207-001M3-11

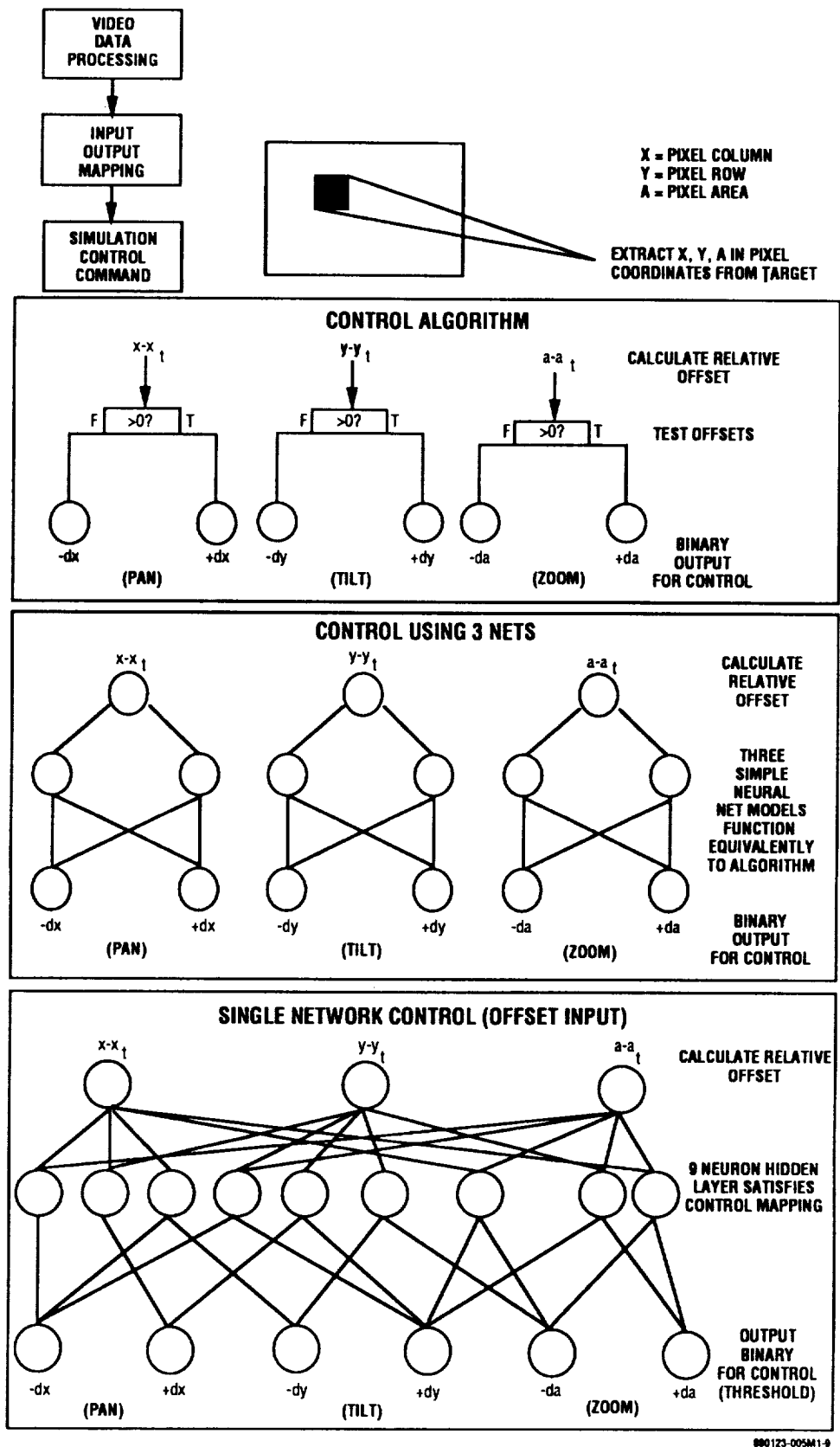
FIGURE 2. SIMULATION PLATFORM BASIC REQUIREMENTS





890209-002M3-11

**FIGURE 3. INTERNAL RESEARCH DEVELOPMENT OF NEURAL TECHNIQUES FOR PERSPECTIVE ADJUSTMENT**



880123-005M1-9

FIGURE 4. THREE PREDICTIVE DISPLAY CALIBRATION CONTROL METHODS WERE DEMONSTRATED

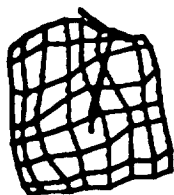
**CYCLES :0**



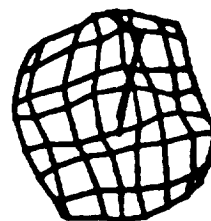
**CYCLES : 166**



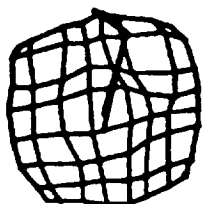
**CYCLES :368**



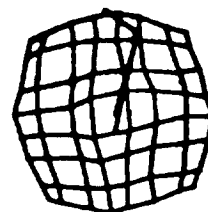
**CYCLES : 877**



**CYCLES : 4085**

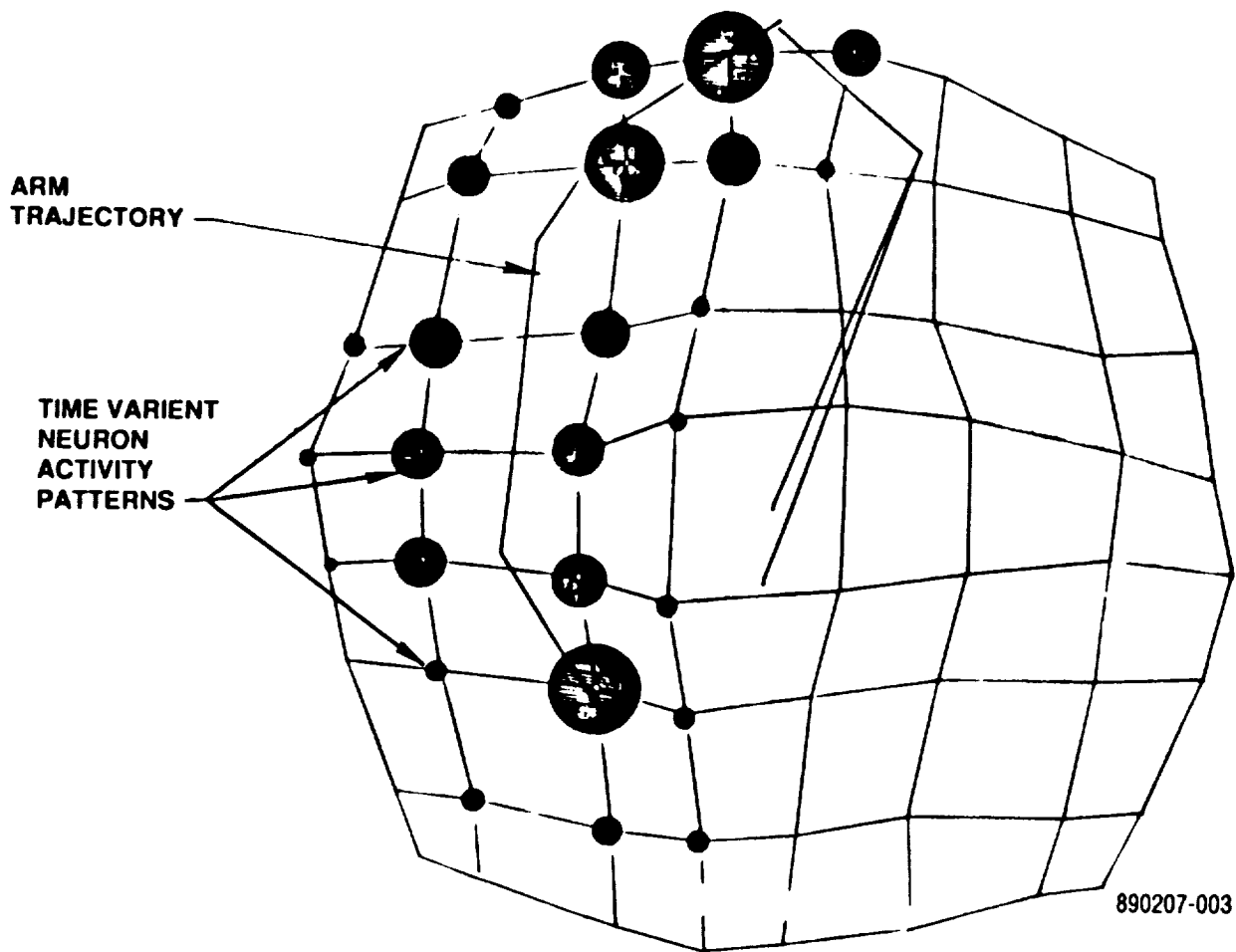


**CYCLES : 27815**



890207-003A

**FIGURE 5. SELF ORGANIZATION FOR MAPPING ARM WORKSPACE/  
PROBLEM DOMAIN**



**FIGURE 6. ACTIVATION PATTERNS IN SELF-ORGANIZED MAPPINGS:  
DYNAMICS MODELING**

## PERFORMANCE EVALUATION OF A 6 AXIS HIGH FIDELITY GENERALIZED FORCE REFLECTING TELEOPERATOR

Blake Hannaford & Laurie Wood  
Jet Propulsion Laboratory  
California Institute of Technology  
Pasadena, CA 91109

### Introduction

Teleoperation is widely expected to perform a wide variety of tasks in future space operations. In order to pursue the goal of a realistic sense of presence at the worksite, many manipulator designs, starting with Goertz (1954), have incorporated force feedback capabilities. As a result, many of the studies attempting to quantify the performance of various teleoperator designs have concentrated on the question "can force/torque feedback improve teleoperation performance?".

Kugath (1972) studied the effects of a compliant manipulation arm and force feedback on manipulation of an inertial load. Results showed force feedback had a large effect on task completion time and error (hitting wall) rate for the maze task. Hill & Salisbury (1977) evaluated three master-slave teleoperators (the Ames Exoskeletal Master Slave, MA11 & MA23) with an instrumented task board which emphasized the peg-in-hole task. Their results documented task completion time for the different arms as a function of peg tolerance and showed improved performance when force feedback was present and reported bare-handed operator performance.

Recent evaluation studies (Draper et.al. 1987) have put emphasis on broadening the base of measurements against which task performance can be judged. Besides task completion time, useful measures included number of task errors, peak force and variance in force (see also Hannaford 1987). Although ANOVA did not show a significant completion time improvement due to force feedback, the other measures did indicating that force feedback allowed the task to be performed with higher quality if not at a faster rate.

The JPL teleoperation Laboratory has recently developed a unique telemanipulation system featuring advanced modes of force feedback and shared control based. This system is described in detail elsewhere in this volume (Szakalay, Kim, Bejczy 1989). The Enhanced 6-Axis Breadboard (ESAB) teleoperation system consists of the JPL-Stanford Force Reflecting Hand Controller (recently refurbished and upgraded from the original [Bejczy & Salisbury, 1981] design), Puma 560 manipulator, and JPL Puma Smart Hand (Fiorini, 1988). The master side of the system is installed in a separate control station without a direct view of the robot work area. Three television cameras provide top, upper left rear, and right rear views of the task board area. The two rear view cameras could be remotely controlled for focus and zoom but fixed views were used for all of the experimental tasks.

The ESAB system can be configured by the user in a wide variety of ways. All control modes and gains can be independently selected for each task space axis. Motion control modes include position control, rate control, and "disabled". The system thus provides a rich set of control possibilities. In this study, five control modes plus direct human task performance were experimentally tested. However, only preliminary experiments were performed with the two modes of shared control. Because of space limitations, this paper only presents three control modes: position control only, position control with force feedback (FFB), and barehanded operator manipulation.

This study is part of a longer term effort to quantitatively evaluate a snapshot of present telemanipulation technology to expose improvements needed for real-world applications. The overall approach was to design a preliminary experiment which looked at a relatively large number of independent variables.

## **Experiment Design**

The experimental design varied control mode, task, and subject. The dependent measures were task completion time, sum of squared forces, and number of errors. Three repetitions of each sub-task were performed by each subject in each of the control modes. All sub-tasks were performed in random sequence to form one repetition (randomization without replacement). All subjects performed all repetitions of a given control mode before the next mode was tested.

The tasks used in experimental teleoperator evaluation fall naturally into two classes: generic tasks and application tasks. Generic tasks are idealized simplified tasks which are designed to test specific telemanipulation capabilities. Application tasks are designed as much as possible to mimic real world uses for teleoperation. Evaluation based on generic tasks illustrates the telerobotics technology push, while application tasks guide the technology in the direction of greatest payoff.

The task board consists of a 21" by 21" frame which accepts modules of either 7" x 7" or 14" x 7". An advantage of the modular task design is that the tasks can be mounted individually on a six-axis force-torque sensor to enable force torque recordings during direct manual operation of the tasks. The four tasks used were: velcro, peg in hole matrix, electrical connectors, and bayonet connector. Each task is in turn broken down into component subtasks:

Task 1 Velcro attachment. Exchange the position of two differently shaped blocks attached to the task board module with velcro. Attempt to attach the blocks securely while minimizing unnecessary force.

Task 2 Peg in hole matrix. this task consists of nine holes arranged in a square matrix. The rows each have a progressively larger clearance, and each column has a different chamfer. The subtasks are to take the standard peg and insert it into a given hole. Peg/hole clearances ranged from 0.005" to 0.0026". Peg diameter was 0.998".

**Task 3 Electrical Connectors.** The subtasks consists of the mating and unmating of three standard electrical connectors: a 3 prong chassis power cord connector, DB25 25pin signal connector, and 1/4" telephone style plug.

**Task 4 Bayonet Connector.** This task consists of unlocking, unmating, mating, and locking a Bendix bayonet style electrical connector (type PT06A-20-16S/16).

Each task was performed according to a pre-specified procedure to which the subjects trained. The task sequences were interspersed with "taps" in which the operator made momentary contact with a designed point on the task board either with the bare gripper, or with a held object. The "taps" injected distinct spikes in the force record (especially the X axis: normal to the task board surface, see Figure 1a). The tasks all began and ended with a tap on a designated square on the task board surface. The resulting force spikes provided well defined benchmarks for measurement and interpretation of the progress of the task by inspection of the force records alone.

Five test operators for this experiment were chosen who would have technical background, but not have in-depth knowledge of robotic technology. Detailed robotic knowledge or knowledge of the specific system itself was felt to be distracting and to not reflect foreseeable operator populations. Subjects recruited were graduate and undergraduate students who were not specialists in robotics. Each subject received 2 to 4 hours of practice on the apparatus. The practice sessions consisted of four, 30 minute sessions in which the task set was performed with and without force feedback.

### Analysis and Performance Measures

The raw force torque data is a rich load of information which can be understood in terms of the task description and which can in turn be used to quantify task performance (Figure 1). This section will describe computations which were performed on the force/torque data to produce performance measures. Completion time, can be determined from the length of the data file containing the force torque data.

Sum of Squared Force (SOSF) is computed by taking a nondecreasing sum of the square of the force or torque values.

$$\text{SOSF} = \sum_i^N f_i^2 dt$$

where  $N$  = number of data samples (task time over  $dt$ ),  $f_i$  is the  $i$ th sample of force or torque, and  $dt$  is the sampling interval (0.01 sec in this experiment). SOSF is accumulated separately for each force and torque axis. A third way in which performance can be measured is through an observer's notations of the "quality" with which a task is performed. In our experiments, a set of "errors" was defined and explained to the test operators and experimenters. A test operator watched each repetition of the experiment and counted occurrences of each error.

In some cases it is desirable to compare performance measures among different segments of the same task. For example, to compare the completion time and SOSF for peg insertion vs. peg extraction. This was accomplished through a computer program

which could recognize benchmarks in the force signal and divide it in time between a set of segments. Returning to the peg-in-hole example (see "x axis force", Figure 1), the data can be clearly divided into "translation" (the manipulator is in free motion: no contact forces), "taps" (sharp spikes in force), "insertion" (predominantly positive forces) and "extraction" (predominantly negative forces). Both "completion time" and SOSF can thus be computed for each segment of the task.

## Results

When experimental records for the peg-in-hole task performed in the several control modes are compared together (Figure 2), the X axis force traces tell most of the story because of the alignment between the task axis and the force/torque sensor's X axis. Comparison of performance in the several control modes shows the reduced completion times and force levels achieved when capability is added to the system.

Although fascinating in themselves, these raw data records are isolated anecdotes of individual task performances. To draw conclusions, the data were reduced to the three basic performance measures. Records of this type for each repetition of each task were processed to produce the performance data points upon which the results below are based. The visually scored error rates were manually correlated with the reduced performance data.

The completion time, SOSF, and number of errors data can be simplified by averaging across one or more dimensions of the design. As a first look at the data, we have computed averages over all subjects and over the first three tasks, "velcro", "peg-in-hole", and "electrical connectors". The fourth task was not included in this average because it took significantly longer than the others (approximately 150 seconds vs. 75 seconds), and was often not completed due to its difficulty. There are 9 subtasks for the peg-in-hole task (corresponding to the 9 test holes) vs. 2 for the velcro and 3 for the electrical connectors. These averages (Figures 3a, 3b, and 3c) show clear trends in performance as the level of capability progresses from position control, through force reflection, up to the bare handed operator.

Completion time (Figure 3a) for the three primary tasks drops from an average of 92 seconds with position control to an average of 63 seconds when force feedback is added. Completion time drops to only 14 seconds for the bare-handed human. SOSF (Figure 3b) drops even more dramatically (from about 3500 to 500 lbs<sup>2</sup> sec) when pure position control is augmented with force feedback and further (to 200 lbs<sup>2</sup> sec) for the bare-handed case.

The number of errors observed (Figure 3c) per repetition drops from 3.0 to 1.1 as force feedback is added. No errors were observed in the bare-handed data.

The probability of the null hypothesis that there was no effect of force feedback (calculated by the two-tailed Z test) was much less than 0.01 in all of the differences reported above giving them a high degree of statistical significance.

These results summarize one of the main results of this study, that the provision of force feedback reduces completion time for a task mix emphasizing energetic interaction and precision manipulation by approximately 30%, reduces SOSF by a factor of 7, and



reduces errors in performing the task by 63%.

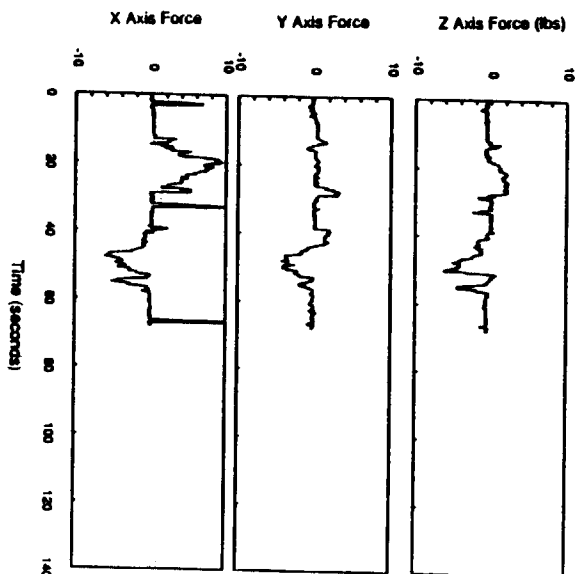
The first level of detail to add to the summary results is to break them down by individual task (Figures 3d, 3e) instead of averaging all the tasks together. Doing this shows that the effect of force feedback is not the same for all of the tasks. For the peg-in-hole task, completion time (Figure 3d) follows the expected course dropping by almost a factor of two from 105 to 59 seconds as force reflection is added. For the velcro blocks task, completion time increases from 72 to 83 seconds. Both of these changes are statistically significant by the method described above. For the electrical connectors, only a slight change is observed which was NOT statistically significant. Of course all of the tasks were completed much faster by the bare-handed operator. The average time in this case is about 15 seconds.

The SOSF data (Figure 3e) tell a different story. As with completion time, for the peg-in-hole task there is a dramatic drop in SOSF (from 5400 to 500  $\text{lbs}^2 \text{ sec}$ ) as force reflection is added. The increase in completion time seen for the velcro task is accompanied by a significant decrease in SOSF (from 800 to 400  $\text{lbs}^2 \text{ sec}$ ). For the electrical connectors, the SOSF measure declines significantly in spite of their unchanged completion time.

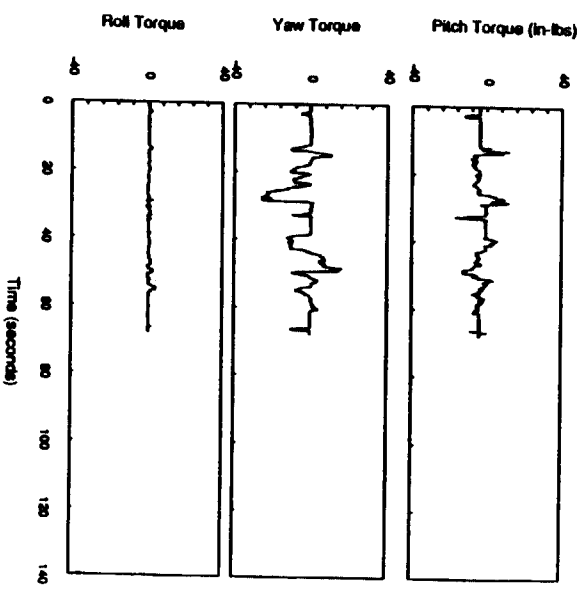
The performance measures were calculated for the segments of 150 repetitions of the peg-in-hole task (Figure 4). Total time (Figure 4a) spent in the movement phase is unchanged at 32 sec by the addition of force feedback. The tap phase is also unchanged at a negligible 2 sec. But the insertion and extraction phases are accomplished markedly faster (31 vs. 11 for insertion, 35 vs. 10 for extraction) when force feedback is present. The two pie charts (Figure 4b,c) illustrate the changing nature of the task mix. As force reflection is added the dominant component of completion time changes from insertion/extraction, the environmental interaction phases, to the free motion phase of the task.

## Conclusions

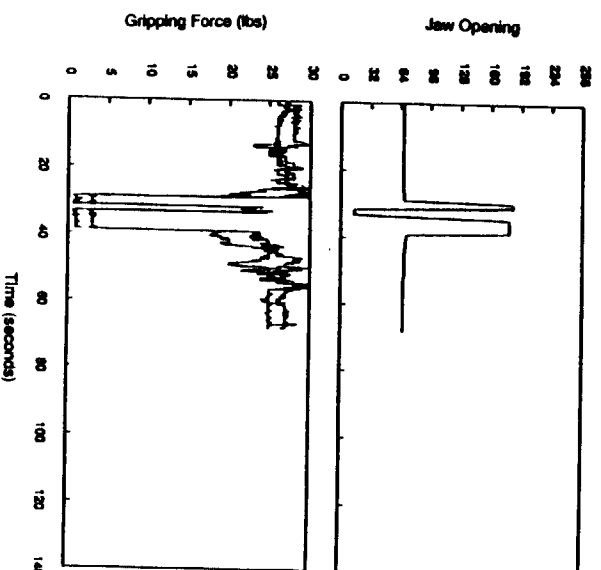
This paper has presented a few of the results of a major study of over 100 hours of experimental teleoperation. Force and torque data recorded from the robot wrist is a rich source of information on the performance of tasks. Performance measures can be computed for whole tasks, or for specific task segments. As a general principle, the performance increases as manipulation capability is increased although the effects may depend on task and performance measure. This study has laid the groundwork for much future work. Further reports will detail additional results which could not be presented here due to lack of space as well as follow-on experiments investigating manipulation under time delay and shared control conditions.



(a)



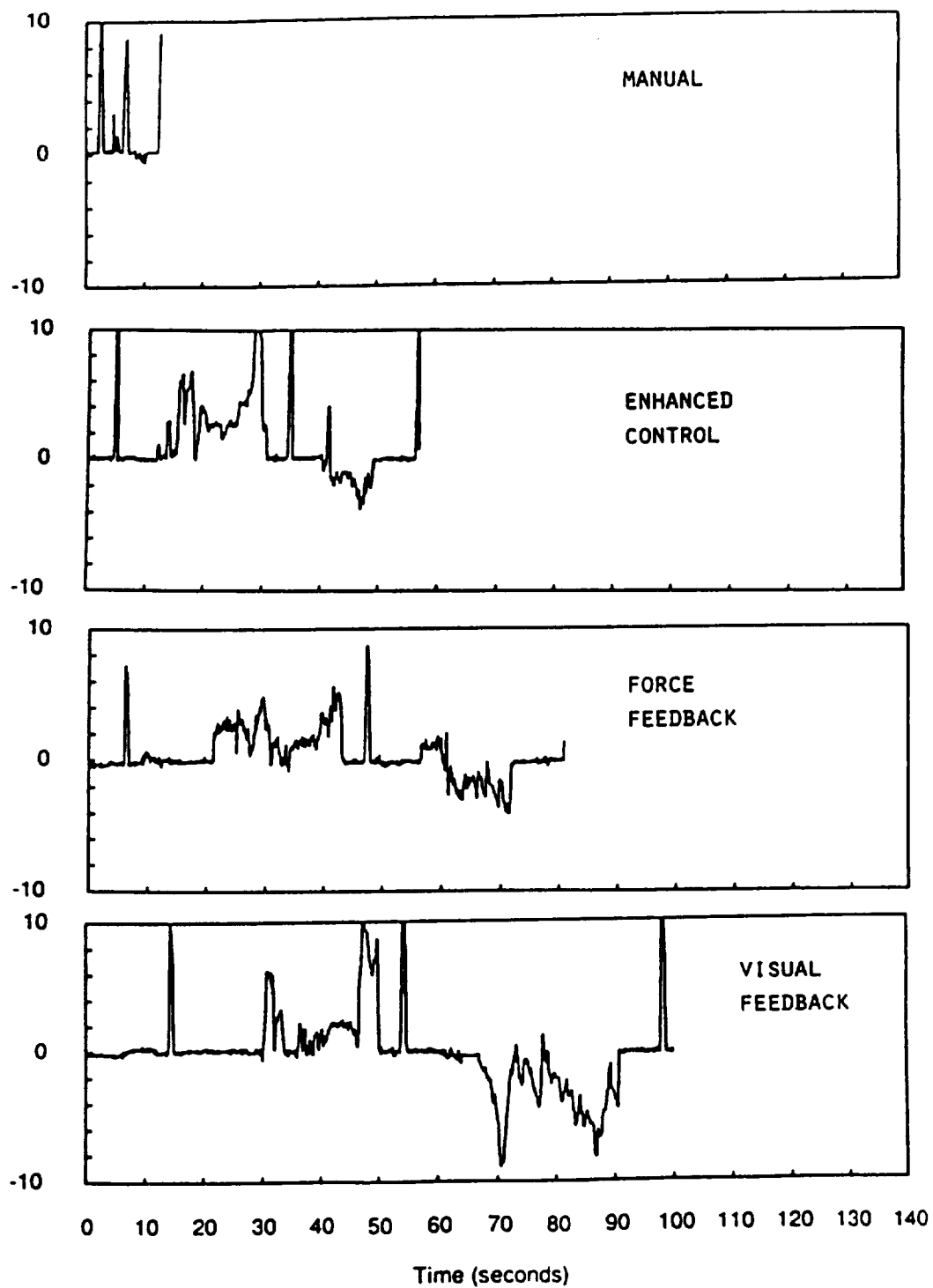
(b)



(c)

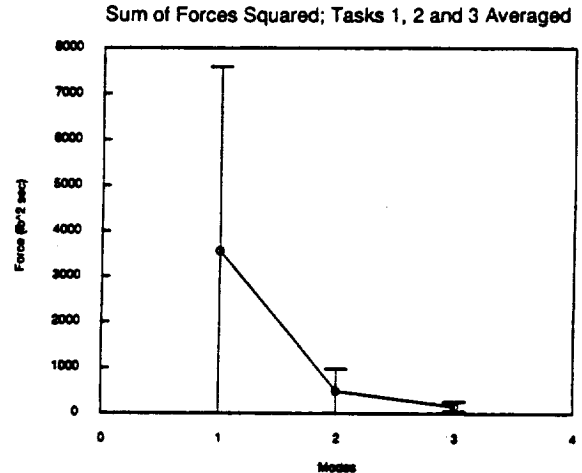
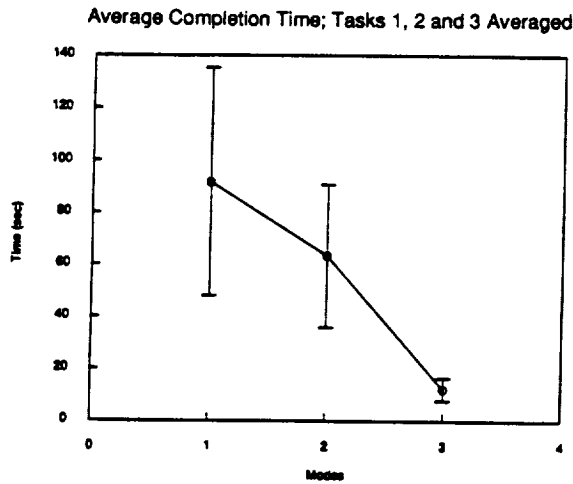
Figure 1

Raw Force Torque Data



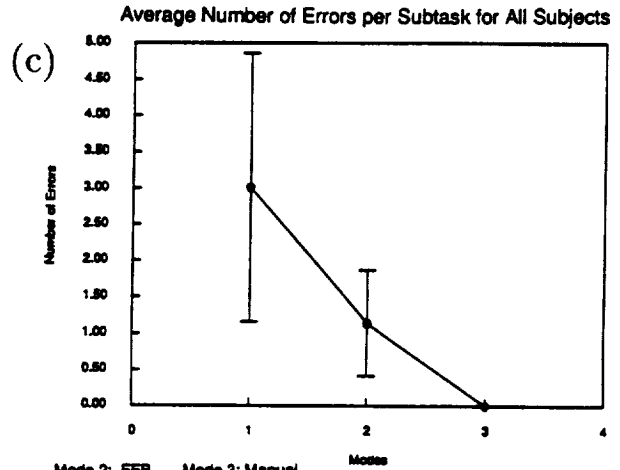
PEG-IN-HOLE #9: X AXIS FORCE

Figure 2



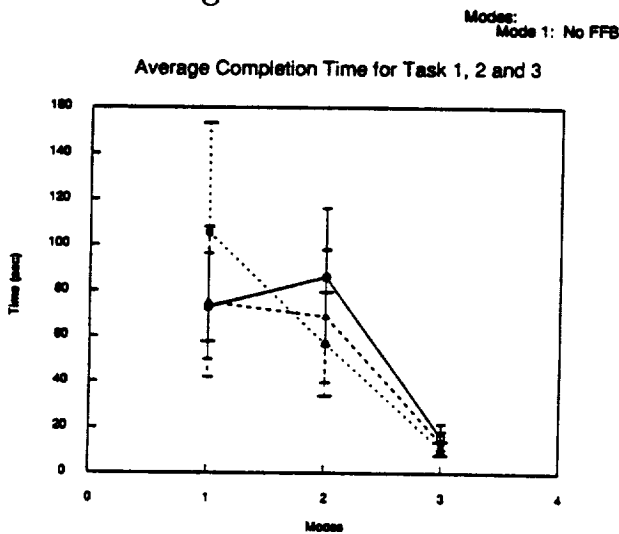
(a) Modes: Mode 1: No FFB Mode 2: FFB Mode 3: Manual  
 Task 1: Velcro Blocks Task 2: Peg in Hole Task 3: Electrical Connectors

(b)

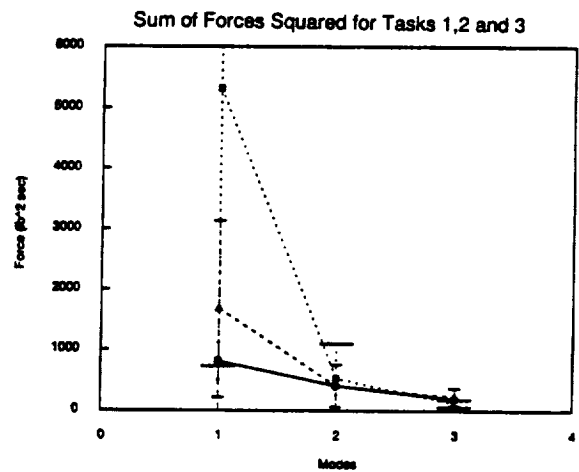


(c)

Figure 3  
 Task Averages

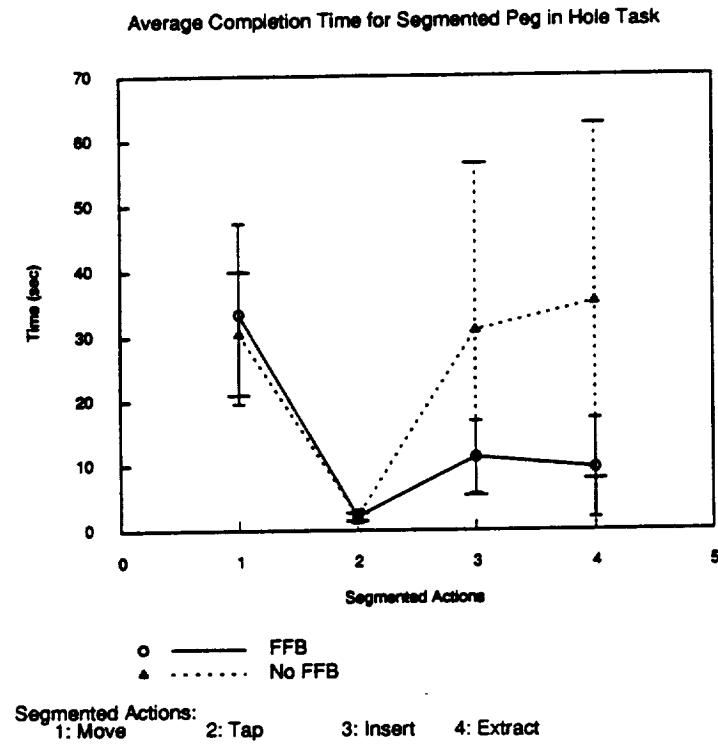


(d)

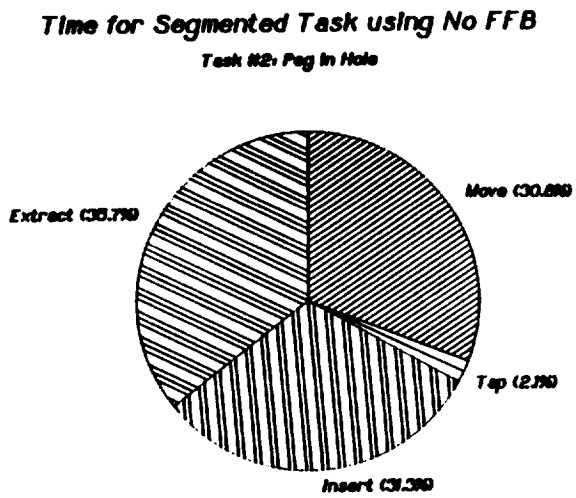


(e)

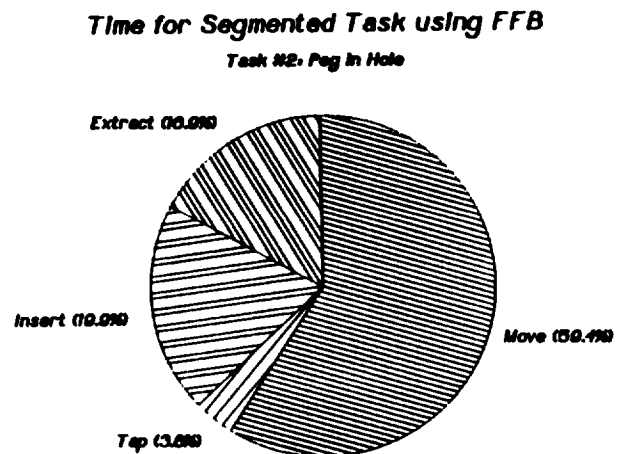
Modes: Mode 1: No FFB Mode 2: FFB Mode 3: Manual



(a)



(b)



(c)

Figure 4  
Segmented Data

## REFERENCES

1. A.K. Bejczy, J.K. Salisbury, Kinesthetic Coupling Between Operator and Remote Manipulator, *Proceedings: ASME International Computer Technology Conference*, 1, San Francisco, CA, Aug. 12-15, 1980, pp. 197-211.
2. J.V. Draper, W.E. Moore, J.N. Herndon, Effects of Force Reflection on Servomanipulator Task Performance, *Proceedings of USDOE/FRG Specialists' Meeting on Remote Systems Technology*, Oak Ridge, TN, June 1-3, 1987.
3. P. Fiorini, A Versatile Hand for Manipulators, *IEEE Control Systems Magazine*, 8, pp. 20-24, 1988.
4. R.C. Goertz, W.M. Thompson, Electronically Controlled Manipulator, *Nucleonics*, pp. 46-47, Nov. 1954.
5. B. Hannaford, Task Level Testing of the JPL-OMV Smart End Effector, *Proceedings of the JPL - NASA Workshop on Space Telerobotics*. JPL Publication 87-13, Vol. 2, pp. 371-380.
6. J.W. Hill, J.K. Salisbury, Study to Design and Develop Remote Manipulator Systems, Annual Report, SRI International, Menlo Park, CA, Nov. 1977.
7. D.A. Kugath, Experiments Evaluating Compliance and Force Feedback Effect on Manipulator Performance, General Electric Corporation, NASA-CR-128605, Philadelphia, Pa, August, 1972.
8. Z. Szakaly, W.S. Kim, A.K. Bejczy, Force-Reflective Teleoperated System with Shared and Compliant Control Capabilities, *Proceedings NASA Conference on Space Telerobotics*: (this proceedings) JPL Publication 89-7, Pasadena, CA, January, 1989.

## Acknowledgments

This research was performed at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration. The authors would like to thank Antal Bejczy, Paolo Fiorini, Daniel Kerrisk, Paul Lee, Derek Parker, and Steven Venema of JPL for their vital assistance.

## IMPLEMENTATION AND DESIGN OF A TELEOPERATION SYSTEM BASED ON A VMEBUS/68020 PIPELINED ARCHITECTURE

Thomas S. Lee

Jet Propulsion Laboratory  
California Institute of Technology  
Pasadena, CA 91109

### Abstract

*This paper describes a pipelined control design and architecture for a force-feedback teleoperation system that is being implemented at the Jet Propulsion Laboratory and will be integrated with the autonomous portion of the testbed to achieve shared control. At the local site, the operator sees real-time force/torque displays and moves two 6-dof force-reflecting hand-controllers as his hands feel the contact force/torques generated at the remote site where the robots interact with the environment. He also uses a graphical user menu to monitor robot states and specify system options. The teleoperation software is written in the C language and runs on MC68020-based processor boards in the VME chassis, which utilizes a real-time operating system; the hardware is configured to realize a four-stage pipeline configuration. The environment is very flexible, such that the system can easily be configured as a stand-alone facility for performing independent research in human factors, force control, and time-delayed systems.*

### Introduction

Many existing teleoperation systems are designed to be purely teleoperative (i.e., to receive input commands solely from the operator). In many robotic applications, it is desirable to mix input commands from the operator as well as from a high level planner to have shared or traded control capability [1]. Space tasks such as bolting a screw on a space station platform need not be performed entirely by the astronaut nor under his continuous supervision. For example, he can perform gross motions such as moving the manipulator to the work vicinity, then trade the mode from teleoperation to autonomous control to allow the machine to complete the task by detecting the bolting location, then invoke compliance control as the bolt is being threaded. This saves the astronaut valuable time since he does not continuously monitor the task as it proceeds. This type of situation is referred to as traded control, for there is no mixture of input modes but a complete turnover of control — the robot is either under human or machine control, not a combination of both. There are many situations however when shared control is desirable or even necessary. One instance is when some form of force control is required. For example, as the robot hand moves along the surface in a window-washing situation, the operator can provide positional setpoints while depending on the machine to provide force control setpoints. In a situation of inserting a replacement module into a satellite, the astronaut can provide the positional information but have the machine provide the orientation information (aligning the module automatically as it is being inserted). In space applications, time-delay introduced in the control loop because of transmission delay can be handled by having a form of shared control (e.g., have the autonomous system at the remote site where the robot is situated handle all internally generated forces during the task and have the user on Earth provide positional commands).

The architecture and design of present day teleoperation systems is such that it is not trivial to incorporate inputs from an autonomous system. The major obstacle is to coordinate inputs (i.e., position or force trajectories) from the teleoperation and autonomous sides and synchronize them to ensure that the

resulting trajectories are consistent. There is need for an effective cooperation between the human and the machine in such a way as to have the human informed of what the machine is doing and vice versa. The Teleautonomous Systems Research Laboratory at JPL has adopted a hardware approach that incorporates various elements of shared control. The design of the teleoperation side was much influenced by experiences gained on previous teleoperation systems built at JPL [2, 3], and the desire was to port to it many concepts that were already proven and demonstrated in the teleoperation laboratory. Another objective was to build a system which has a flexible hardware and software development environment that can easily accommodate various modes of shared control in position, orientation, force, and torque domains, obtaining commands from the human operator and/or the autonomous system.

This paper is organized into 5 sections and an appendix. Section 1 provides a description of theoretical aspects of how force-feedback teleoperation is achieved. Section 2 describes the hardware and software environment that exists in the laboratory. In Section 3, the details of pipeline implementation are elaborated, especially the aspects that pertain to timing. Section 4 describes the user interface and how it is achieved. The main text of the paper is concluded in Section 5.

## 1. Overview of Teleoperation Concepts

The teleoperation system accommodates various modes of operation: joint, Cartesian, rate, index, and force-feedback modes. Joint mode is implemented by having a one-to-one mapping of each hand controller's degrees of freedom (DOF) to that of the robot — this mode is used to test hardware interfaces and to move the robot out of kinematic singular positions. Scaling is involved since the angular ranges of the robot and those of the hand controller are not equivalent. Cartesian mode is when the operator moves the robot in position control mode, having the end-effector referenced with respect to the robot tool frame or the defined world frame. There is a one-to-one correspondence between the motion of the operator's hand and the motion of the robot end-effector. If the robot is in rate mode (either Cartesian or joint space), the robot speed is controlled relative to the amount of deflection of the hand controller handle from its initial start-up (neutral) position. In this mode, the hand controller is in a "spring-return" state such that if the handle is released the hand controller will return to the neutral position (the effect is analogous to a spring-return joystick). Index mode allows the user to extend the workspace of the hand controller. Once a bound of a certain hand controller joint is reached, the user presses the index button to inform the system to disregard hand controller input (i.e., not to move the robot). He then moves the hand controller away from the joint bound and presses the index button to reactivate the robot. Finally, force-feedback is a mode that allows the operator to feel on his hand the forces/torques that are generated at the tool tip of the robot as it interacts with the environment.

The following paragraph explains the theory of operation when the system is under Cartesian control mode. Refer to Figure 1. The forward loop of the teleoperation system during each sampling interval is the path from the hand controller sending incremental trajectory information to the remote site, which directs the robot to move as commanded by the operator moving the hand controller. Positional information is relayed from the local to the remote site by sending incremental  $\Delta X$  information. This Cartesian  $\Delta X$  information is computed by premultiplying the  $\Delta\theta$  angular values with the hand controller Jacobian expressed with respect to the tool (or base frame). Once the  $\Delta X$  information is received at the remote site, it is transformed to be expressed with respect to the robot tool (or base frame). The transformation matrix has the following form:

$${}^R J_H = \begin{bmatrix} R & \vdots & 0 \\ \dots\dots\dots \\ 0 & \vdots & R \end{bmatrix}$$

Typically this transformation is a constant  $6 \times 6$  rotational matrix and accounts for the orientational difference between the hand controller and the robot. In a space environment, the robot base may have a moving orientation (e.g., the space platform where the robot is placed may move with respect to the shuttle where



the hand controller is placed) — the  $6 \times 6$  transformation matrix would not be constant in this case. The  $\Delta\theta$  is calculated for the robot by premultiplying the transformed  $\Delta X_R$  with the computed inverse Jacobian of the robot. Indexing is accomplished by sending  $\Delta X_H = 0$  from the local to the remote site. At times, the robot may drift (i.e., absolute position error will exist between the hand controller and robot positions due to accumulation of  $\theta$  differentiation (linearization) error). This position error is very difficult to notice since the operator is using teleoperation to move the robot in a relative sense and does not keep track of the ideal robot position). In our system, this error is not handled because we have a high sampling rate and noise-free  $\Delta\theta$  data (obtained by differencing two successive encoder values of the hand controller rather than from a velocity approximator); however, one can add an absolute position servo loop at the robot side to account for the error.

The feedback loop originates from the sensor attached at the tip of the robot, where interaction forces and torques are felt, and this sensory information is sent to the local site and reflected onto the hand controller by backdriving the motors that cause the operator's hands to feel the encountered forces and torques. The sensed force/torque information is sent from the remote site and is received at the local site by using the same parallel interface used to pass the  $\Delta X_H$  information from the local to the remote site. Desired torques to be applied to the hand controller motors are computed as follows. First, the force/torque values from the sensor frame are transformed to the robot tool frame and then premultiplied by a  $6 \times 6$  diagonal matrix to scale and to express reaction forces and torques to be felt by the human. Finally, the forces and torques are premultiplied by the transpose of the hand controller Jacobian express torques that are applied to the motors of the hand controller.

The transformation matrix that converts the force/torque sensor data to resolved Cartesian components expressed with respect to the hand controller handle frame has the following form:

$${}^H J_{Sensor} = {}^0 J_6^T \text{ Reaction and Scaling } J_H \text{ Tool } J_{Sensor}$$

where

$${}^0 J_6^T = \text{Hand Controller Jacobian Transpose}^1$$

$$\text{Reaction and Scaling } J_H = \text{diag}(k_1, k_2, k_3, k_4, k_5, k_6)$$

$$\text{Tool } J_{Sensor} = \begin{bmatrix} R & \vdots & 0 \\ \dots\dots\dots \\ p \times R & \vdots & R \end{bmatrix}$$

where

$$p \times = \begin{bmatrix} 0 & -p_z & p_y \\ p_z & 0 & -p_x \\ -p_y & p_x & 0 \end{bmatrix}$$

**Handling Singularity.** When a robot is in a singular position, there are multiple kinematic solutions. For example, in the case of a PUMA 560 robot, when joints 4 and 6 axis become aligned, a degree of freedom is lost and in the kinematic sense, only the sum of joint 4 and joint 6 is then important. Therefore, in this situation, the user would have to specify either the joint 4 or 6 value to force the inverse kinematic solution to be unique. When a typical teleoperation scenario is considered, if one observes that the robot is moving toward its singular position, then this indicates two possible actions by the operator. One option is that the operator wants to change the robot pose, and the other is that he has made a mistake by moving the robot near the singular position. This ambiguity can easily be resolved by querying the operator. In our system, the operator specifies his intention to change pose by pressing the middle button when the robot is near a particular singularity. If the operator does not press the pose change button near the singular position,

<sup>1</sup>Refer to the Appendix for the hand controller (JPL's FRHC) kinematic model.

then the robot is forced to remain in the existing pose (i.e., the robot is prevented from ever going into the singularity – a “bouncing off” effect). Pose change can only occur near the singular regions, rather than anywhere in the robot workspace, to avoid large swinging motions.

## 2. System Descriptions

**Hardware.** The hardware is divided between two sites: the local and the remote. Refer to Figure 2 for the hardware configuration. The operator located at the local site moves both the right and the left hand controllers with his hands and observes the corresponding robots located at the remote site moving according to his hand motions. At the local site, the operator moves two six DOF universal Force-Reflecting Hand Controllers (FRHC) [4]. The FRHC that is integrated into the system is the third generation hand controller (version C) built in-house at JPL; it is capable of generating 8 lbs of force and 14 in-lbs of torque in each DOF. The design is such that a six-axis mechanism with a steel-cable/pulley drive system is used to virtually eliminate backlash; various miniature ball bearings and large diameter pulleys were used to reduce mechanism friction. Each axis is driven by DC torque motors with digital incremental encoders for feedback information such as position and velocity. The point of contact for the operator on the FRHC is the handgrip. It is used to specify orientation information while the task is proceeding, and the operator can use three momentary buttons to change system operating modes. The top-left button is called the index button and is used to activate the robot — the operator usually turns the index button off when he interacts with the system menu. The top-right button is used to turn on/off force reflection while a task is proceeding. The middle-bottom button is to confirm a robot pose change. Finally, the trigger is used to open/close the robot gripper.

The hardware unit that interfaces with either the FRHC or the PUMA robot is called the Universal Motor Controller (UMC) [5]. It was built in-house at JPL and can be easily reconfigured to interface with either a hand controller or a robot. It contains joint interface cards that provide encoder and potentiometer information and output desired PWM signals, and uses two National Semiconductor 32016 CPU boards to execute servo and communication software written in the NSC32016 assembly language. Communication between the processor boards and the joint interface cards is made through the Multibus. A special set of protocols to communicate with the UMC from the VME side was developed and tested; the rate of communication is approximately 2 KHz, which is twice the rate at which internal PD motor servoing is performed. A parallel port on one of the CPU boards handles communication and is used to interface with the VME side. The UMC-VME interface allows on-line capability of specifying desired encoder setpoints (position commands) or torque commands (analogous to the PUMA Unimation controller's current commands) in PWM units, reading actual encoder positions and other analog signals (in the case of a hand controller, reading the trigger potentiometer value), setting control loop gains (for position and velocity), controlling the robot gripper, and calibrating the robot or the hand controller.

The VME chassis environment (either L-TELEOP or R-TELEOP) at the local site consists of the following: four MC68020/68881 processor boards (Heurikon HK68/V2F) each running at 20 MHz clock rate, to perform robot/hand controller kinematic computations, communication, graphics, and network interfacing with the user (in the future, to reduce kinematic computation time, a MC68030/68882 Heurikon HK68/V30 card will be used); an Ethernet card (Excelan EXOS 202) to connect to the local network; a system controller card (Motorola MVME025) that handles bus arbitration; a graphics generator card (Parallax 600 VME) for force/torque displays; and two parallel communication cards (Xycom XVME-240) for interfacing with the UMC and the remote site. The graphics card generates real-time force, torque, and grasp force information for operator display.

In the remote site, the hardware includes two Unimation PUMA 560 robot arms. Each arm is equipped with a commercial wrist force torque sensor from the Lord Corporation and a TRI servo gripper. The arms are driven by two UMC's. The VME chassis environment at the remote site consists of the following: five MC68020/68881 cards — two of them perform kinematics for the right and left robots, two perform communication with the right and left robots, and the last one performs network interfacing; a bus arbiter; four parallel communication cards — two interface with the robots and the other two with the Lord

force/torque sensor processor units; an Ethernet card; and a shared memory bus adapter card from the BIT3 Corporation to obtain shared control commands from the autonomous portion of the testbed (SUN 4 running the RCCL robot language under dual-arm configuration).

One of the convenient features of the JPL architecture is the homogenous hardware environment. The VMEbus/68000 architecture was chosen in part to be compatible with the SUN computer backplane environment, which uses the VMEbus architecture — the code for the autonomous portion executes on the SUN4 and an interface exists between the SUN4 and the robot VME chassis by utilizing a VME-VME bus adapter. The choice of the VME architecture seems to be popular, since many research centers have now adopted the architecture for their robotics research. At the motor control level, the local and remote UMC's and VME environments have identical hardware and software setups (i.e., processor and interface cards have the same hardware configurations, and the same code that can handle either the local or remote site is downloaded and executed). Having a homogeneous environment has a number of advantages, namely that the system can be reconfigured easily for many different types of research, and the same resources such as robots and controllers can be shared. Elements of redundancy in the hardware add to fault protection and reliability of the system.

**Software.** All teleoperation software is written in the C language with the exception of the NSC32016 assembly language code that runs on the UMC. Code is developed on a SUN 3/60 (or SUN 4 with a cross compiler) UNIX computer utilizing SUN's C compiler and Wind River's VxWorks/Wind real-time library and is downloaded through Ethernet to the processor boards for immediate execution. Many convenient features of the VxWorks library such as task control, networking, and debugging support save a great deal of development time.

### 3. Pipeline Architecture Implementation and Timing Data

In this section, a four-stage pipeline design is described. From the hardware point of view, pipeline architecture can be considered modular since functionalities are divided among the processor boards. Referring to the pipeline diagram of Figure 3, each processor has a unique assigned function (e.g., two of the processors  $COM_H$  and  $COM_R$  are dedicated solely to handling communication, and the other two  $KIN_H$  and  $KIN_R$  perform kinematic computations). Considering modularity, for a  $KIN$  board, the computation time required to perform the assigned function can be reduced by replacing the board with a faster processor board, and as a result, since the most time-consuming ( $KIN$ ) stage has been speeded up (in pipeline design, the most time-consuming stage determines the pipeline clock period), the pipeline is executed faster. Another consideration is that due to the modular design, available processing power is optimally utilized. Processor assignment can be made according to the required computational power for each stage. For example, a  $COM$  stage that requires little number-crunching capability can be assigned to a processor board that holds minimal computation power (enough to handle handshaking with its communicating partner), while a  $KIN$  stage should be assigned to a fast processor board that is equipped with an auxiliary floating point processor running at optimal clock rate.

Pipeline design does not increase the closed-loop control loop delay (computation time) but does increase the throughput of the system (i.e., system sampling rate is increased due to the increased rate at which input data is gathered and output data is generated). It is not certain however that pipeline design results in added system stability or performance. Conventionally the sampling and computation times are set equal. In this case, all effects occurring between sampling instances will not be detected. But these effects between each successive sampling will be noted and compensated for if multiple sampling was made during each computation period. It is important to consider the transient effects, especially if an anomaly condition occurs — the faster the system senses the anomaly, the faster the system will respond to it. In this sense, it is intuitive that the higher the sampling rate, the more responsive the system will be in providing more effective control actions.

Various studies have been made concerning time-delay in a force-reflection teleoperation system [6, 7] which is related to lengthening the closed-loop control loop delay in the system. Solutions such as providing

local compliant force control at the remote site and robot-positional-error feedback to reduce instability caused by time-delay have been presented. However, this type of analysis does not address the advantage of having a pipeline design. Pipeline design touches on the issue of multirate sampling theories. Various digital control texts describe sampling concepts. Shannon's sampling theorem [10] states that the original sampled signal can be reconstructed by having a sampling rate that is at least twice the bandwidth of the cutoff frequency of the system. Due to the effects of noise, data quantization, and system resonant frequencies, in their discussion of Shannon's sampling theorem, Houps and Lamont [10], recommend that the sampling rate be at least eight times greater than the bandwidth of the reference input. Craig [9] considers noise and resonant effects and recommends that the sampling rate be 10 times faster than the correlation time of noise or of the structural resonant frequency of the manipulator mechanism. Avoiding structural resonance is also discussed in Paul [11]. He recommends that the sampling rate should be 15 times the link structural frequency. All these arguments favor having a fast sampling time for the manipulator. In the present design, a four-stage pipeline architecture will be implemented to have the sampling rate be approximately 6 times faster than the closed control loop delay time in force-reflection mode. No conclusive evidence was found by the author as to the advantage of having a faster sampling but still retaining the same closed-loop delay time. Using our configuration, this issue will be studied further, and in connection the effects of multirate sampling in robotic systems will be investigated.

The details of the pipeline design will now be presented. Figure 3 shows a timing diagram and lists the actual timing data that were obtained after implementation. To perform teleoperation computations, four processors are coordinated in a pipeline arrangement; each stage of the pipeline is handled by one processor. Each processor at every 1.6-ms period performs its designated computations (see description boxes underneath the timing diagram). For example,  $COM_H$  stage starts by communicating with the robot side to exchange  $\Delta X$  and force information for 0.3 ms, multiplies the transpose of the hand controller Jacobian to force values (in 0.1 ms), communicates with the UMC to send desired torques and at the same time to receive the present encoder positions, and finally stays idle for 0.6 ms until the next 1.6-ms period begins. The diagram contains a shaded path that traces the closed-loop force control flow. The control loop begins by having the  $COM_H$  stage receive the UMC encoder values — this requires 0.6-ms communication time.  $KIN_H$  stage then takes the converted robot angular values and calculates the hand controller Jacobian to compute  $\Delta X$  values. Stages  $COM_H$  and  $COM_R$  synchronously get invoked to pass the  $\Delta X$  information to the robot site. Transformation is made to express the  $\Delta X$  information with respect to the robot base frame, and then the inverse Jacobian of the robot is multiplied to compute the desired robot positional setpoints. After waiting for the force sensor to respond to the robot servoing to the desired setpoints (which is approximately 1.6 ms — usually the Lord force/torque sensor processing unit sends resolved Cartesian force/torque information at every 10 ms, but we are using the raw strain gauge mode with increased clock speed to obtain data much faster), 0.3 ms is used to obtain the raw strain gauge values and multiply these readings with a sensor calibration matrix to compute corresponding force/torque values. The force/torque values are then forwarded to the hand controller side, and finally forces are converted to desired torques and sent to the hand controller UMC for torque servoing. The closed-loop sampling is approximately 104 Hz. In implementation, the multiple processors are synchronized and data is passed through shared memory. In a pipeline situation, the stage that has the worst time delay dictates the pipeline clock rate. In our design, the most time-consuming stages are the kinematics stages  $KIN_H$  and  $KIN_R$ , which take around 1.6 ms. In the future, each of these stages will be replaced by a faster processor (MC68030/68882) board which has a faster clock speed and floating point capability and which will increase the pipeline clock rate. Note that it is more difficult to improve timing for the communication stages; since the processors do little number-crunching but are used to synchronize the communication protocol, upgrading these processors will not improve the timing.

#### 4. Operator Interface

A user interface has been developed using the TCP/IP communication protocol on Ethernet, which allows the operator to execute the teleoperation software and specify system options from any remotely located computer that supports the protocol. For the operator, a graphics menu is displayed on a SUN 3/60 terminal, through which he can interact with the system. The menu consists of a number of windows whose

implementation is based on the SUNVIEW facilities [8]. It is a user-friendly environment where choosing an option can be done simply by moving the SUN mouse and then clicking on a button. It is capable of displaying graphics information in different formats (e.g. bars and scales to display robot information and icons to represent various parts of the system). The operator can use the menu to specify options such as system control modes (position with or without force feedback, and rate, joint, and shared control modes) and reference and view frames. He can monitor the state of the robot by observing the robot angles on the menu display and monitor force/torque data displays on another monitor — observing the robot angles is useful in detecting joint limits. For each robot, a menu window is dedicated for displaying the data about that particular robot. See Figure 4. The robot data is forwarded to the SUN for menu display through a specially designed protocol based on UNIX socket facilities. This data is not forwarded at every sampling instance of the system, but once every tenth or more sampling time, which is sufficient to display varying real-time data. In addition, Cartesian position is forwarded as well as the present robot configuration. Since the data is available on the SUN computer, robot and force/torque data can easily be logged for later analysis.

## 5. Conclusions

In this paper, a force-feedback teleoperation system based on a pipeline architecture was described. It will be integrated with the autonomous portion of the JPL testbed to support shared control research. Once the system is operational, issues such as multirate sampling effects will be investigated. Future work will include extensive experimentation with the system to examine control, human factors, and time-delay issues.

## 6. Acknowledgements

The research described in this document was performed at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration. The author would like to thank Samad Hayati for many useful discussions and Ted Lewis and Zoltan Szakaly for their software and hardware support.

## References

- [1] Hayati, S., Venkataraman, S.T., "Design and Implementation of a Robot Control System with Traded and Shared Control Capability," submitted for publication to the *Proceedings of the 1989 IEEE International Conference on Robotics and Automation*
- [2] Bejczy, A.K., and Hannaford, B., "Man-Machine Interaction in Space Telerobotics," *Proceedings Intl. Symposium on Teleoperation and Control*, Bristol, England, July 1988.
- [3] Bejczy, A.K., Salisbury, J.K., "Controlling Remote Manipulators Through Kinesthetic Coupling," *Computers in Mechanical Engineering*, Vol. 2, No. 1, July 1983, pp. 48-60.
- [4] McAfee, D., Ohm, T., "Teleoperator Subsystem/Telerobot Demonstrator: Force Reflecting Hand Controller Equipment Manual," JPL D-5172 (internal document), Jet Propulsion Laboratory, Pasadena, California, January 1988.
- [5] Bejczy, A.K., and Z. Szakaly, "Universal Computer Control System for Space Telerobotics," *Proc. of the IEEE Conference on Robotics and Automation*, Vol. 1, pp. 318-324, Raleigh, N.C., 1987.
- [6] Anderson, R.J., and Spong, M.W., "Bilateral Control of Teleoperators with Time Delay," *Proc. IEEE Int'l. Conf. Systems, Man, and Cybernetics*, Vol. 1, p. 131, Beijing, China, August 1988.
- [7] Hannaford, B., "Stability and Performance Tradeoffs in Bilateral Teleoperation," submitted to the *Proceedings of the IEEE Intl. Conf. on Robotics and Automation*, Scottsdale, AZ, May 1989.
- [8] Sun Microsystems, Inc., SunView System Programmer's Guide, September 1986.

- [9] Craig, J., *Introduction to Robotics*, Addison-Wesley Publishing, 1986, pp. 239-242.
- [10] Houpis, C. H., Lamont, G. B., *Digital Control Systems*, McGraw-Hill, 1985.
- [11] Paul, R., *Robot Manipulators: Mathematics, Programming, and Control*, The MIT Press, 1981.

## 7. Appendix: JPL Force-Reflecting Hand Controller Kinematic Model

The Denavit-Hartenberg parameters for the FRHC are given below:

$\theta_i$	$\theta_1$	$\theta_2$	$90^\circ$	$\theta_4$	$\theta_5$	$\theta_6$
$d_i$	0	0	$d_3 + d_{3\_offset}$	0	0	0
$a_i$	0	0	0	0	0	0
$\alpha_i$	0	$90^\circ$	$-90^\circ$	0	$-90^\circ$	$90^\circ$
Initial Settings	0	$-90^\circ$	$d_3 = 0^\circ$	$90^\circ$	$-90^\circ$	$0^\circ$

$d_{3\_offset} = 730.25mm$

$$\begin{aligned}
 {}^0A_1 &= \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} ; \quad {}^1A_2 = \begin{bmatrix} c_2 & -s_2 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ s_2 & c_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
 {}^2A_3 &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & d_3 + d_{3\_offset} \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} ; \quad {}^3A_4 = \begin{bmatrix} c_4 & -s_4 & -0 & 0 \\ s_4 & c_4 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
 {}^4A_5 &= \begin{bmatrix} c_5 & -s_5 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -s_5 & -c_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} ; \quad {}^5A_6 = \begin{bmatrix} c_6 & -s_6 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ s_6 & c_6 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
 \end{aligned}$$

Jacobian for the JPL Force Reflecting Hand Controller:

$${}^6J_6 = \begin{bmatrix} j_{11} & j_{12} & j_{13} & 0 & 0 & 0 \\ j_{21} & j_{22} & j_{23} & 0 & 0 & 0 \\ j_{31} & j_{32} & j_{33} & 0 & 0 & 0 \\ j_{41} & j_{42} & 0 & j_{44} & j_{45} & 0 \\ j_{51} & j_{52} & 0 & j_{54} & j_{55} & 0 \\ j_{61} & j_{62} & 0 & j_{64} & 0 & 0 \end{bmatrix}$$

$$r_{30} = d_3 + d_{3\_offset}$$

$$t_1 = c_2 c_4 s_5$$

$$j_{11} = r_{30}(t_1 c_6 - c_2 s_4 s_6)$$

$$j_{21} = -r_{30}(t_1 s_6 + c_2 s_4 c_6)$$

$$j_{31} = -r_{30}(c_2 c_4 c_5)$$

$$t_2 = c_2 s_4 s_5 + s_2 c_5$$

$$j_{41} = c_2 c_4 s_6 + c_6 t_2$$

$$j_{51} = c_2 c_4 c_6 - s_6 t_2$$

$$j_{61} = s_2 s_5 - c_2 s_4 c_5$$

$$j_{12} = r_{30}(c_4 s_6 + s_4 s_5 c_6)$$

$$j_{22} = r_{30}(c_4 c_6 - s_4 s_5 s_6)$$

$$j_{32} = -r_{30} s_4 c_5$$

$$j_{42} = s_4 s_6 - c_4 s_5 c_6$$

$$\begin{aligned}
j_{32} &= s_4 c_6 + c_4 s_5 s_6 \\
j_{62} &= c_4 c_5 \\
j_{44} &= j_{13} = c_5 c_6 \\
j_{54} &= j_{23} = -c_5 s_6 \\
j_{64} &= j_{33} = s_5 \\
j_{45} &= s_6 \\
j_{55} &= c_6
\end{aligned}$$

## FIGURES

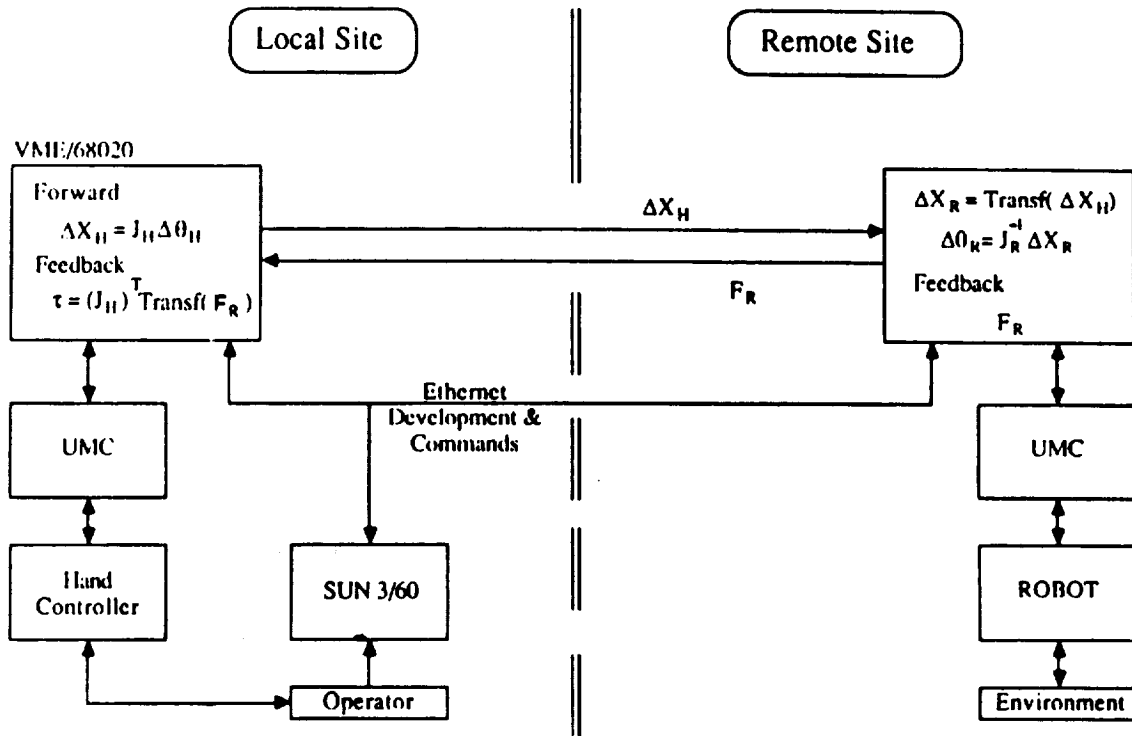


Figure 1: Functional Diagram of the Teleoperation System

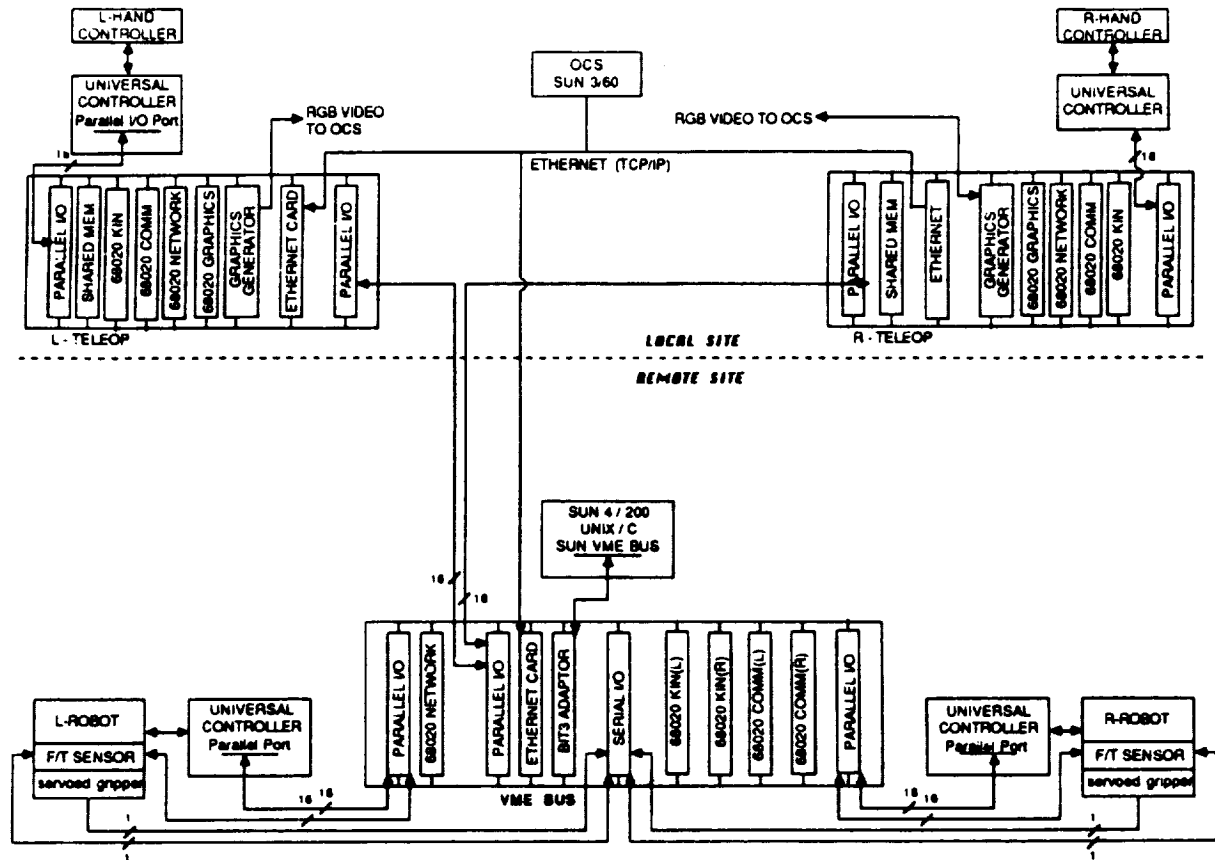


Figure 2: Hardware Diagram of the Teleoperation System

ORIGINAL PAGE IS  
OF POOR QUALITY



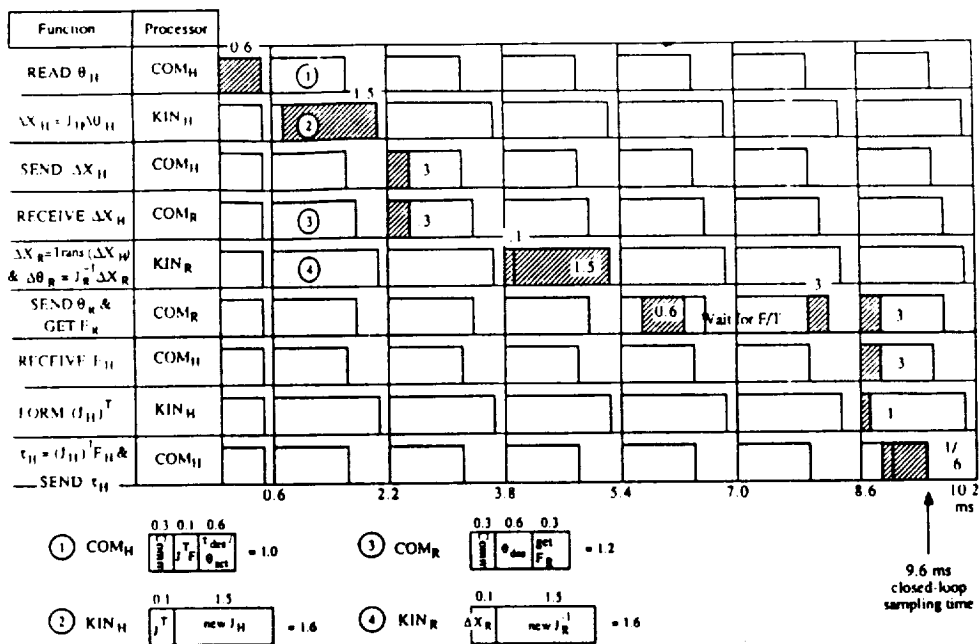


Figure 3: Timing Diagram

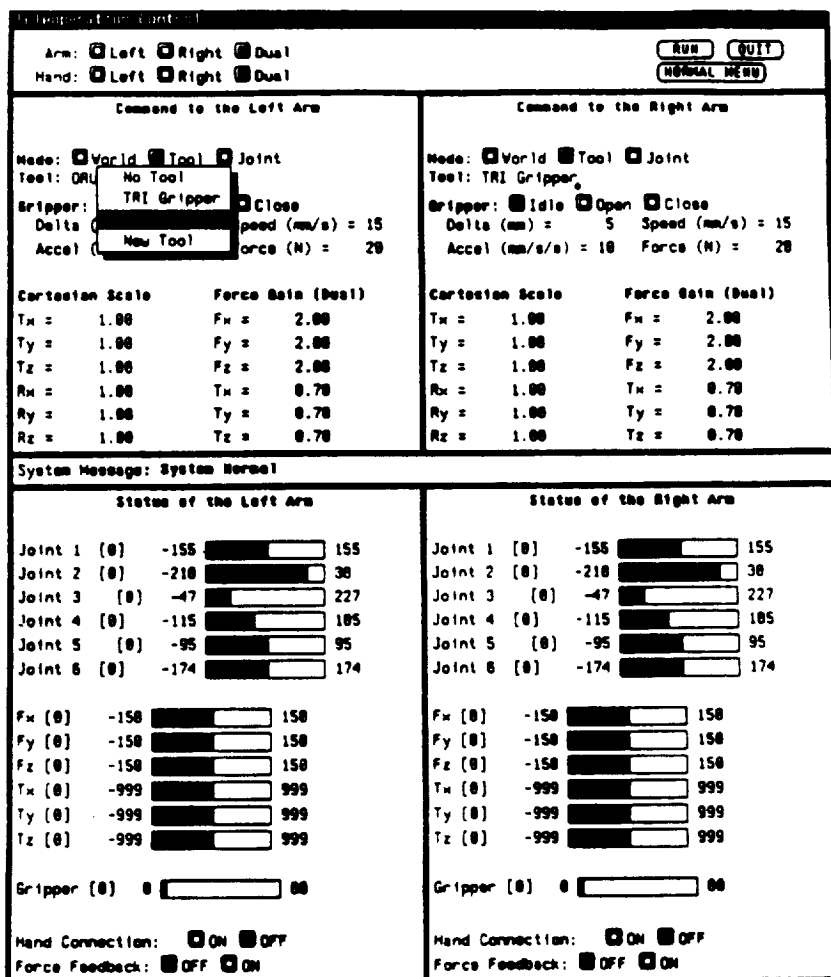


Figure 4: Graphics User Menu



## Human Machine Interaction via the Transfer of Power and Information Signals

H. Kazerooni, W. K. Foslien, B. J. Anderson, T. M. Hessburg

Mechanical Engineering Department  
University of Minnesota  
Minneapolis, MN 55455 USA

### Abstract

Robot manipulators are designed to perform tasks which would otherwise be executed by a human operator. No manipulator can even approach the speed and accuracy with which humans execute these tasks. But manipulators have the capability to exceed human ability in one particular area: strength. Through any reasonable observation and experience, the human's ability to perform a variety of physical tasks is limited not by his<sup>1</sup> intelligence, but by his physical strength. If, in the appropriate environment, we can more closely integrate the mechanical power of a machine with intellectually driven human hand under the supervisory control of the human's intellect, we will then have a system which is superior to a loosely-integrated combination of a human and his fully automated robot as in the present day robotic systems. We must therefore develop a fundamental approach to the problem of this "extending" human mechanical power in certain environments. "Extenders" will be a class of robots worn by humans to increase human mechanical ability, while the wearer's intellect remains the central intelligent control system for manipulating the extender. The human body, in physical contact with the extender, exchanges information signals and power with the extender.

Commands are transferred to the extender via the contact forces between the wearer and the extender as opposed to use of joystick (master arm), push-button or key-board to execute such commands that were used in previous man amplifiers. Instead, the operator becomes an integral part of the extender while executing the task. In this unique configuration the mechanical power transfer between the human and extender occurs in addition to information signal transfer. When the wearer uses the extender to touch and manipulate an object, the extender transfers to the wearer's hand, in feedback fashion, a scaled-down value of the actual external load which the extender is manipulating. This natural feedback force on the wearer's hand allows him to "feel" the scaled-down value of the external forces in the manipulations. Extenders can be utilized to maneuver very heavy loads in factories, shipyards, airports, and construction sites. In some instances, for example, extenders can replace forklifts. This article describes the experimental results for a prototype extender<sup>2</sup>.

### 1. Introduction

Manipulators have the potential to exceed human ability in one particular area, strength. The ability of a human to lift heavy objects is determined by his own muscular strength. The ability of a robot manipulator to perform the same tasks depends upon the available actuator torque. A relatively small hydraulic actuator can supply a large torque. In contrast, the muscular strength of the average human is quite limited. Extenders will be a class of robot manipulators which will extend the strength of the human arm, while maintaining human control of the task. The extender is distinguished from conventional master-slave<sup>3</sup> systems; the extender is worn by

<sup>1</sup> The pronouns "he" and "his" used throughout this article are not meant to be gender-specific.

<sup>2</sup> For the general analysis on extender dynamics and control, contact H. Kazerooni at the above address.

<sup>3</sup> A master-slave system (tele-operator system) uses a control joystick of similar geometry to the manipulator for input. The joystick has position transducers at the joints to measure displacement, and the output from these transducers is used as an input to the manipulator. Thus the motion of the manipulator follows that of the joystick. The joystick is called the master

the human for the purpose of direct transfer of power. Consequently, there is actual physical contact between the extender and the human, allowing transfer of mechanical power in addition to information signals<sup>4</sup>. Because of this unique interface, control of the extender trajectory can be accomplished without any type of joystick, keyboard, or master-slave system. The human provides an intelligent control system to the extender, while the actuators ensure most of the necessary strength to perform the task. The key point is the concept of "transmission of power and information signals". The human becomes a part of the extender, and "feels" some scaled version of the load that the extender is carrying. In contrast, in a conventional master-slave system, the human operator may be either at a remote location or close to the slave manipulator, but he is not in direct physical contact with the slave in the sense of transfer of power. Thus the operator can exchange information signals with the slave, but mechanical power is not exchanged directly. In a typical master-slave system, natural force reflection does not occur because the human and the slave manipulator are not in direct physical contact. Instead, a separate set of actuators are required on the master to reflect forces felt by the slave back to the human operator<sup>5</sup>.

In the extender system, the input to the extender will be derived from the set of contact forces resulting from the contact between the extender and the human. This set of contact forces is being used to manipulate an object in addition to generating information signals for the extender control. Note that force reflection occurs naturally in the extender; the human arm will feel a scaled down version of the actual forces on the extender without a separate set of actuators. For example, if an extender is used to manipulate a 100 lbf object, the human may feel 10 lbf while the extender will take the rest of the load. The 10 lbf contact force is used not only for manipulation of the object, but also for generating the appropriate signals to the extender controller. In other words, the contact force between the human and the extender is measured, appropriately modified (in the sense of control theory to satisfy the performance and stability), and used as an input to the extender control, in addition to being used for actual maneuvering.

A simple example is given in Figure 1a to show some fundamental concepts about the extender. Figure 1a shows a one degree of freedom extender, moving a load. If the load weight is  $W$ , at equilibrium, the following equality is true for the extender. (Figure 1b)

$$\tau + f_h h = W l \quad (1)$$

where  $\tau$  is the actuator torque and  $f_h$  is the force imposed by the human on the extender. The goal is to develop a control algorithm in the system such that  $f_h h$  is always a constant portion of  $\tau$ . In other words, the human always feels a scaled down version of the actual necessary force to lift the load. Suppose the load weighs 100 pounds, while  $l=2'$  and  $h=1'$ , it is then desired to control the extender such that  $f_h=10$  lbf, for example, while  $\tau=190$  lbf.ft. Note that the 10 lbf on the extender, imposed by human, is the amount of force that is used to help lifting the load. The human will feel this 10 lbf as a reaction force (toward down in Figure 1). The human uses this force as a natural reflection to feel the scaled down version of the actual force. If the system is accelerating, the total load in lifting  $W$  with acceleration of  $\dot{v}_e$  and velocity of  $v_e$  is  $[W l \sin(\theta) + J \dot{v}_e]$  where  $J$  is the moment of the inertia of the extender and load. ( $\theta$  is measured from a vertical line).

$$\tau + f_h h = W l \sin(\theta) + J \dot{v}_e \quad (2)$$

A control algorithm must be designed such that  $f_h h$  is constant and a small portion of  $\tau$ .

manipulator, and the mechanical manipulator is called the slave. Ideally, the motion of the slave will be identical to that of the master.

<sup>4</sup>The human-machine interaction in active systems has been traditionally characterized by the exchange of "information signals" only. For example in human-computer interaction, the human sends information signals to the computer via a keyboard. In another example, a car driver sends an information signal to the engine by pushing the accelerator. There is no power transformation between the driver and the car; the driver does not feel the load on the car.

<sup>5</sup> The elimination of force feedback in remote master-slave manipulation may result in poor positioning precision and possible instability [18, 25].

## 2. History and Background

The extender employs a direct physical contact between the human and the manipulator for the purpose of accepting power and information signals. The concept of a device to increase the strength of a human operator using a master-slave system has existed since the early 1960s. The concept was originally given the name "man-amplifier". The man amplifier was defined as a type of manipulator which has the effect of greatly increasing the strength of a human operator, while maintaining human supervisory control of the manipulator. Note that previous systems were designed based upon the master-slave concept, rather than the direct physical contact between human and manipulator for the purpose of power and information signals [4, 8,9,10,11,17,20,21, 22].

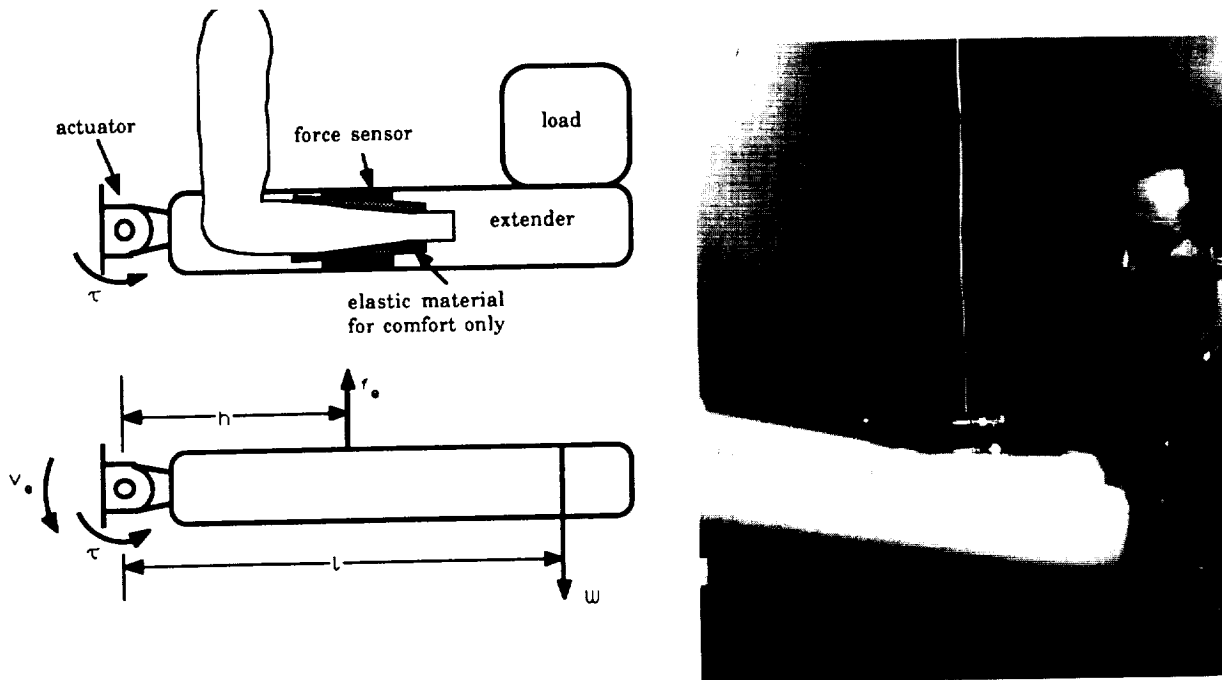


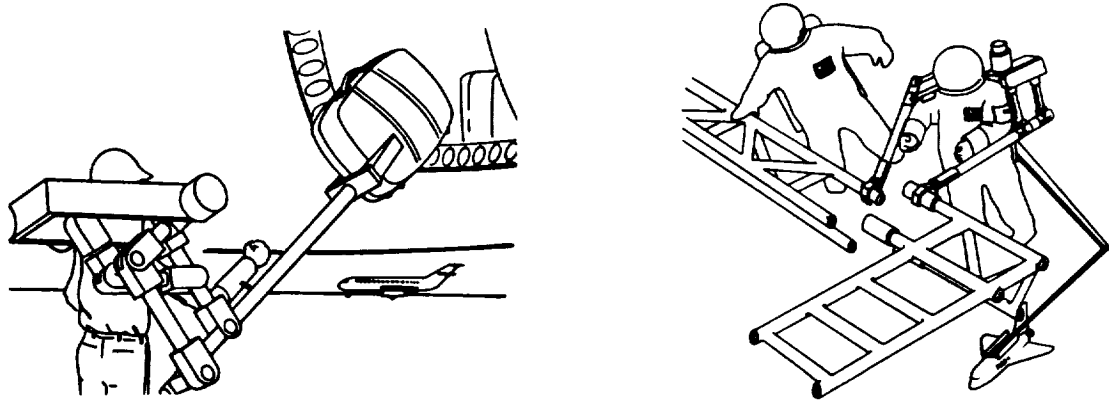
Figure 1: a: One degree of freedom (dof) experimental extender. b: The free body diagram of the extender. c: The experimental one dof extender at the University of Minnesota. This experimental extender is made of steel (160 lbf) to simulate the load.

In contrast with the Hardiman and other man amplifiers, the extender is not a master-slave system. There is no joystick or master device for information transfer. Instead, the human operators commands to the extender are taken directly from the interaction force between the human and the extender. This interaction force is also used to help the extender manipulate an object. In other words, the power and information signals transfer simultaneously at one point. The controller developed for the extender translates the signals representing the interaction force signals into a motion command for the extender. This allows the human to initiate tracking commands to the extender in a very natural way<sup>6</sup>.

<sup>6</sup> A point must be made about what we mean by "natural way". If "talking" is defined as a natural method of communication between two people, then we would like to communicate with a computer by talking rather than using a keyboard. The same is true here; if we define "maneuvering the hands" as a natural method of moving loads, then we would like to only move our hands to maneuver a load, as opposed to using any keyboard or joystick.

Some of the major areas of application for the extender might include manufacturing, construction, loading and unloading aircraft, maneuvering cargo in shipyards, foundries, mining or any situation which requires precise and complex movement of heavy objects. Two main categories of manipulation have been defined for the extender: constrained and unconstrained. In unconstrained maneuvers, the extender is free to move in all directions without any interaction with another system. On a factory floor where heavy objects need to be moved about, the extender could be worn by a worker who would then have the ability to lift and carry these objects. This would be an example of unconstrained maneuvering. Currently, heavy pieces may be moved about by forklifts, pulleys, cranes or similar equipment. The extender will offer an advantage over these methods because it is designed to follow the human arm motions in a very "natural" way. The human will be able to manipulate heavy objects more easily without the use of any key board, joy stick or push button. It is expected that the human operator will be able to maneuver heavy loads with greater dexterity, speed, and precision. In comparison with existing systems such as forklifts, pulleys, and cranes, the extender offers the human the opportunity to adjust the orientation of objects. Figure 2 shows the schematic of the architecture for a prototype multi-dof extender being built at the University of Minnesota. This type of motion may be required for manipulating cargo in a shipyard, assembly tasks, or in a construction application such as installing large windows. The extender is shown without a base for clarity. In reality, the extender might be attached to a mobile or stationary base. Also note that the sleeve into which the human's arm would be inserted is eliminated in the interest of clarity.

The second category of manipulation with the extender is constrained manipulation. This type of manipulation includes any movement which requires interaction with a third object, the "environment". Examples of constrained manipulation by the extender might include operation of a pneumatic jack, bending of materials, or press fitting.



**Figure 2: The schematic representations of the prototype extender, being built at the University of Minnesota.**

The extender also has the potential to become a useful upper limb orthosis for the physically impaired. An orthosis is an externally applied device which improves the functionality of an impaired limb<sup>7</sup>. The main purpose of an orthosis is to enhance the functionality of existing body segments, in contrast with a prosthesis, which serves to replace body segments [2,3,5,23, and 24].

The extender would be classified as an orthosis, rather than a prosthesis, because it would enhance existing motor ability instead of replacing an absent segment. The extender would augment the lifting ability of the patient and also allow continued use of the patient's remaining motor ability. For a patient to employ the extender, he must have some ability to move his arm.

<sup>7</sup>Appropriate modification of the extender for this use would include decreasing the overall size of the extender, decreasing the size of the actuators used, and improving the cosmetic appearance of the extender. Recent discoveries in superconductivity may lead to design and construction of electric motors with high power to weight ratio so they can be employed to power the extender.

The capability for some motion is necessary because the extender requires motion from the user in order to move. Thus, the patient must use his remaining muscle ability to drive the extender. The extender would serve to improve the patient's limb function while utilizing the remaining natural limb function.

### 3. Experimental Extender

To understand the issues in control and dynamics involved in human/machine interaction, the control of an experimental one dof extender is described (Figure 1c). The general building blocks on nonlinear dynamics and control (in particular the stability of the human and extender taken as a whole) are given in references 7 and 11. Figure 3 shows the schematic of the control loop for a one dof experimental extender. Two forces add up to maneuver the extender:  $f_e$  and  $\tau$ . The contact force between the human and the extender,  $f_e$ , is the result of human intention to move up the extender and the actuator torque,  $\tau$ , is the result of the feedback. A velocity controller is chosen as the lowest level of control for the extender so the extender is stabilized independently of the human dynamic behavior<sup>8</sup>.

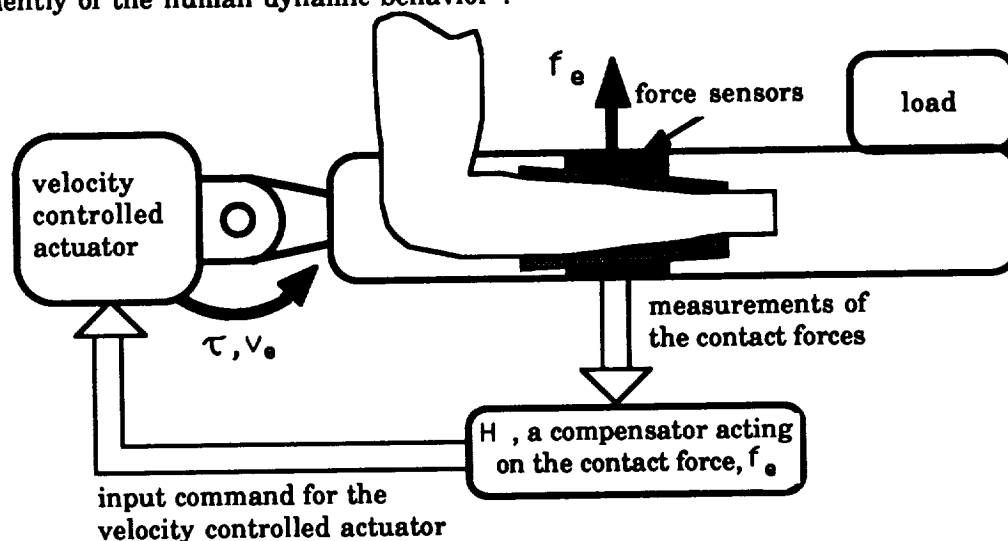


Figure 3: The schematic of the one dof extender.  $f_e$  is the force imposed on the extender by the human.  $\tau$  and  $v_e$  are the torque and the velocity of the extender.

The interaction force between the human and the extender is simply fed back and used (after passing through the compensator,  $H$ ) as an input to the velocity controlled extender. When the human pushes against the extender, the contact force,  $f_e$ , is measured and passed through the compensator,  $H$ . The output of this compensator is used as the input command for the velocity controlled actuators of the extender. When the human does not push against the extender, the contact force,  $f_e$ , and consequently the input command to the actuator are zero. The zero command for the velocity controlled actuators results in zero speed for the extender. In other words, when there is no push from the human, the extender will be stationary.  $H$  is of paramount importance in the stability of the system of the human and the extender taken as a whole<sup>9</sup>. For a given load, it is desirable to have the bandwidth of the extender wide so it can keep up with the high speed motion of the human arm. It is also desirable to have the contact force remain as small as possible so one

<sup>8</sup> It is of practical importance that the extender be stable when the human is not wearing it.

<sup>9</sup> Similar analysis is given in references 15 and 16 to describe the stability of an *autonomous* robot interacting with an environment.

can maneuver a large load with a small contact force<sup>10</sup>. It has been shown in [7] and [11] that in order to achieve a fast response and a small (but nonzero) contact force one needs large values for  $H$ . However, one cannot choose an arbitrarily large value for  $H$ ; the stability of the system must also be guaranteed. References 7 and 11 describe the instability via a formal mathematical framework. Here it is explained how instability may occur in the system when a large value for  $H$  is chosen. Suppose the compensator  $H$  has a large gain<sup>11</sup> over a frequency range of operation. If the human decides to move up the object, the extender will move up with such a large velocity that it pulls the human arm up. This reverses the direction of the contact force between the human and the extender (downward in Figure 3). Then the extender responds to the downward force with a large velocity which will pull down the human arm. This periodic motion occurs in a very short amount of time and the motion of the extender will become oscillatory and unbounded.  $H$  must be designed such that its gain is large enough for the human to maneuver an object with high speed while stability is guaranteed.

First, the dynamic behavior of the experimental 1 dof extender and its velocity controller<sup>12</sup> is given here. An explanation of how one additional force feedback passing through a compensator allowing for a stable interaction will follow. The prototype extender is powered by an EXCELLO SS-8-100 limited rotation hydraulic actuator (100° total rotation, 1800 ft.lbf maximum torque at 3000 psi). A MOOG 72-102 2-stage servovalve has been used to drive the actuator. The servovalve has the rated flow of 40 GPM at 1000 psi, with 0.02 Amps of the input current. The dynamic behavior of a servo hydraulic actuator is governed by equations 3-5. Equation 3 is the valve dynamics while equations 4 and 5 represent the flow continuity and actuator dynamics [19].

$$Q_l = K_q I - K_p P_l \quad (3)$$

$$Q_l = v_e D_m + \frac{V_t}{4\beta_e} \frac{d}{dt} P_l \quad (4)$$

$$P_l D_m = J \dot{v}_e \quad (5)$$

where:

$Q_l$  = load flow (in<sup>3</sup>/sec)

$K_q$  = flow gain (7700 in<sup>3</sup>/sec/Amp for MOOG 72-102, 2-stage servovalve)

$I$  = current to drive the servovalve

$K_p$  = pressure gain

$v_e$  = angular velocity of the extender (rad/sec)

$D_m$  = actuator volumetric displacement (7.62 in<sup>3</sup>/rad for EXCELLO SS-8-100)

$J$  = moment of inertia of the extender in Figure 3 (113.6 in.lbf.sec<sup>2</sup>)

$\beta_e$  = hydraulic fluid modulus of elasticity (100,000 psi)

$V_t$  = total contained volume in actuator (13.3 in<sup>3</sup> for EXCELLO SS-8-100)

combining equations 3-5, equation 6 will result as an open loop transfer function that maps the servovalve input current to the extender velocity.

<sup>10</sup>The contact force should be small but non-zero. It is necessary to have non-zero contact force, so the human always feels a constant portion of the actual load.

<sup>11</sup> One can use the singular value for linear systems or  $L_p$  norm for nonlinear systems to represent the gain.

<sup>12</sup> The nature of the velocity controller is not of importance in this analysis. One can always use a number of advanced nonlinear control methodologies for the development of robust velocity controllers for robotic applications [26, 27]. In the simplest case, one can design a velocity controller for each degree of freedom of the extender independently, while satisfying the extender closed loop stability.



$$G_p(s) = \frac{v_e}{i} = \frac{\frac{K_q}{D_m}}{\frac{s^2}{\omega_e^2} + \frac{2\zeta_e s}{\omega_e} + 1} \quad (6)$$

where  $\omega_e$  and  $\zeta_e$  are given by the following equations:

$$\omega_e = \sqrt{\frac{4\beta_e D_m^2}{V_t J}}, \quad \zeta_e = \frac{K_p}{D_m} \sqrt{\frac{\beta_e J}{V_t}}$$

$K_q/D_m$  is a nonlinear function of the pressure drop across the valve, the load on the actuator, and the distance that the valve is stroked away from null.  $\zeta_e$  is highly nonlinear, and will increase rapidly past unity as the valve amplitude is increased. The theoretical value of  $\omega_e$  in the neighborhood of the operating is 11.8 hertz<sup>13</sup>. The theoretical open loop transfer function (equation 6) was then compared to experimental frequency response to find actual value for  $\omega_e$ ,  $\zeta_e$  and  $K_q/D_m$ . Experimental verification of the actuator dynamics was performed by driving the system with a sinusoidal signal and observing the velocity output from the tachometer. Figure 5 shows the experimental frequency response of the open loop system. The experimental transfer function results in a damping ratio  $\zeta_e = .45$ , a hydraulic natural frequency  $\omega_e = 8.4$  hertz, and a plant gain  $K_q/D_m = 220$  rad/sec/Amp. Compensator  $K(s)$  is then designed to develop a closed loop velocity control for the extender (Figure 4). Equation 7 shows the proposed transfer function for the compensator,  $K(s)$ . The integrator overcomes the friction forces and the lead compensators generate positive phase angle for the loop transfer function for stability. Proposing equation 7 for the compensator, the closed loop transfer function is given by equation 8.

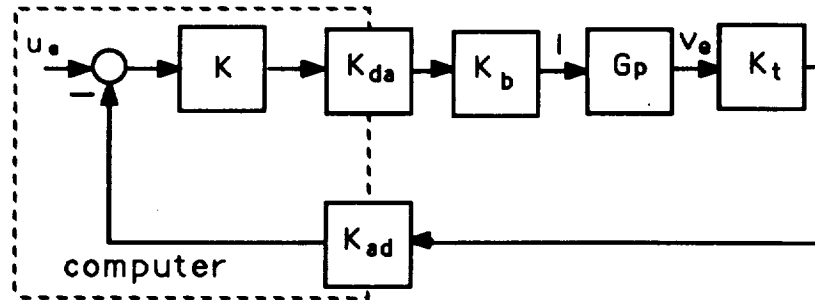


Figure 4: The Closed Loop Velocity Control.  $u_e$  is the input velocity command from the computer. The arguments of the transfer functions have been eliminated in all the block diagrams.  $K_{da}$ : D/A convertor gain(10 Volts / 2048),  $K_b$ : Servocontroller board gain (.0077 Ampere/Volts),  $K_t$ : tachometer gain(.5Volts/rad/sec) ,  $K_{ad}$ : A/D convertor gain (2048 / 1.25 Volts)

$$K(s) = K_o \frac{(\frac{s}{\alpha} + 1)(\frac{s}{\beta} + 1)}{s} \quad (7)$$

$$G_e(s) = \frac{v_e}{u_e} = \frac{K_o K_{da} K_b \frac{K_q}{D_m} (\frac{s}{\alpha} + 1)(\frac{s}{\beta} + 1)}{(\frac{1}{\omega_e^2})s^3 + (\frac{2\zeta_e}{\omega_e} + \frac{\gamma}{\alpha\beta})s^2 + (1 + \frac{\gamma}{\alpha} + \frac{\gamma}{\beta})s + \gamma} \quad (8)$$

where:

$$\gamma = K_o K_{da} K_b \frac{K_q}{D_m} K_t K_{ad} \quad (9)$$

<sup>13</sup>This number includes Meritt's 40% reduction factor [19, page 140].

$\alpha = 90$  rad/sec,  $\beta = 100$  rad/sec, and  $K_o = 1.6$  allow for the widest bandwidth for the closed loop velocity control. This bandwidth is limited by the high frequency unmodeled dynamics in the system [12,13, and 14]. The experimental and theoretical dimensionless closed loop frequency response plots (figure 6) show a bandwidth of approximately 10 rad/sec (1.7 hertz).

The next level of control involves the design of a compensator that operates on the contact force between the extender and the human. The emphasis of the human arm model is on the functional relationship between the dynamic input and output properties of the human arm. Therefore, there is less concern about the internal structure of the components in the model. The particular dynamics of nerve conduction, muscle contraction and central nervous system processing are implicitly accounted for in constructing the dynamic model of the human arm. With regard to the above assumption two variables affect the human arm trajectory: 1) the commanded trajectory issued from the human central nervous system,  $u_h$ , and 2) the external force on the human arm imposed by the extender,  $f_h$ . The integration of the above two dynamical properties results in the dynamic equations of the human arm.

$$y_h = G_h(u_h) + S_h(f_h) \quad (10)$$

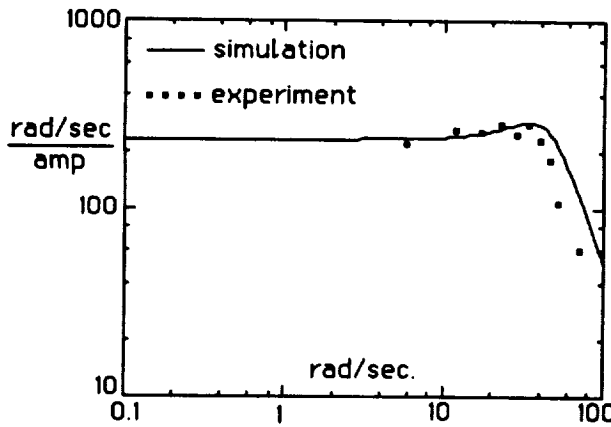


Figure 5: The Frequency Plot of the Open Loop Extender,  $G_p(s)$

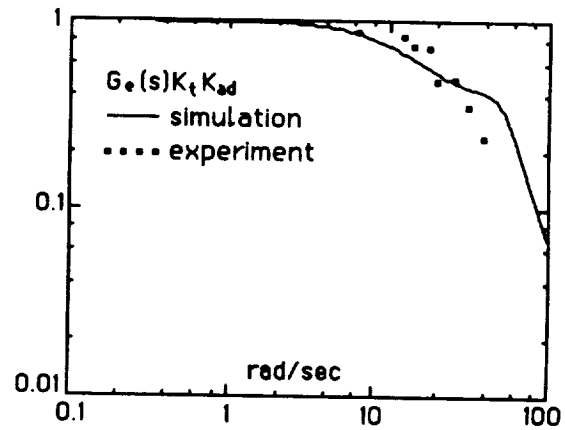


Figure 6: The Dimensionless Frequency Plot of the Closed Loop Velocity Dynamic Behavior

Whenever a force is applied to the human arm, the end-point of the human arm will move in response. The sensitivity function  $S_h$ , is defined as a mapping from the imposed forces,  $f_h$ , on the hand to the resulting displacement of the human hand. In the simplest case, one can think of  $S_h$  as the reciprocal of the hand muscles.  $G_h$  represents the mapping from commanded trajectory issued from the human central nervous system to the human hand position,  $y_h$ .  $G_h$  and  $S_h$  are generally nonlinear mappings; however in this example they can be considered as transfer functions that map  $u_h$  and  $f_h$  to  $y_h$ . Figure 7 shows the basic structure for the closed loop control system of the one dof experimental extender.  $E$  represents the physical compliance of the human arm flesh and the force sensor which is located between the human arm and the extender. Since the force sensor is very stiff,  $E$  will be dominated by the physical compliance of the flesh. Force sensor amplifier gain,  $K_f$ , translates the contact force to a voltage, which is then fed into the computer.

The transfer function for the position of the extender is as follows:

$$\frac{y_e}{u_h} = \frac{G_e H K_f E G_h K_{ad}}{G_e H K_f E K_{ad} + s(1 + E S_h)} \quad (11)$$

From equation 11, the larger  $H$  is chosen to be, the closer  $y_e$  will be to  $G_h u_h$  and in the limit when  $H \rightarrow \infty$  then  $y_e \rightarrow G_h u_h$  (the extender will follow the human command perfectly). However one

cannot choose an arbitrarily large value for  $H$ ; stability of the system in Figure 7 must also be guaranteed. Raising the gain of  $H$  will increase the extender closed loop bandwidth until a point is reached where the extender can no longer be operated in a stable manner. The linear stability condition is given by inequality 12. If one guarantees the condition<sup>14</sup>, then the system will remain stable; however if one does not satisfy inequality 12, no conclusion can be made. On the other hand, if the system is unstable, then inequality 12 must have been violated.

$$|H| < \left| \frac{s}{G_e K_f K_{ad}} \left( \frac{1}{E} + S_h \right) \right| \quad (12)$$

The above stability condition does not directly depend on the internal structure of the variables; one can use various transfer functions for  $G_e$ ,  $S_h$  or  $E$  with different orders in inequality 12. The compensator,  $H$ , was chosen as a first order filter in order to reject high frequency components of the command signal which could adversely affect system stability and performance.

$$H = \frac{K_h}{\tau s + 1} \quad \tau = .05 \text{ sec} \quad (13)$$

Since inequality 12 is only a sufficient condition for stability, violation of this condition does not lead to any conclusion. It was observed experimentally that the closed loop system remains stable for all  $K_h < 0.6$ . Figures 8 and 9 show two stable cases where the extender velocity,  $v_e$ , is proportional with the extender input,  $u_e$ . ( $u_e$  is plotted with the velocity unit as  $u_e/K_f K_{ad}$ ; this allows for dimensionless ratio for these two variables which is consistent with the plot of Figure 6.) Figure 10 shows an experiment with  $K_h = 1.7$  where the system becomes unstable and oscillates. Figure 11 shows that the stability criteria has been violated for  $K_h = 1.7$ . This shows the sufficiency of the stability condition.

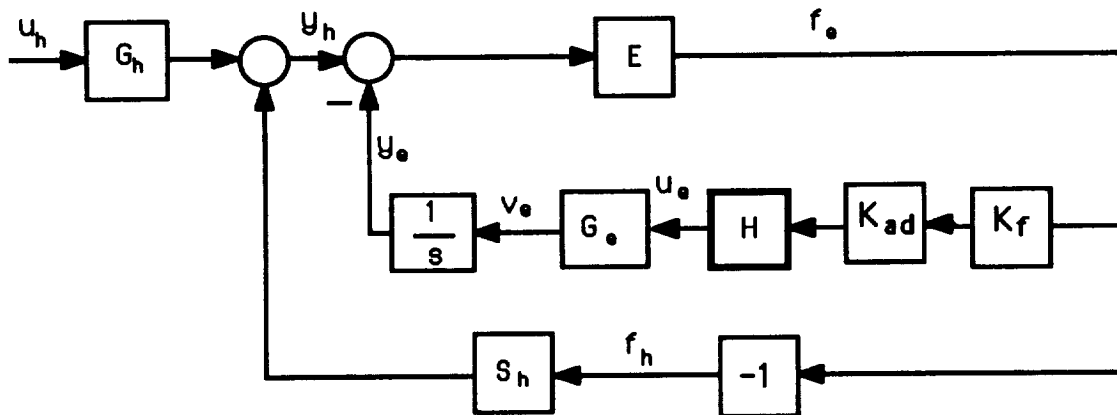


Figure 7: The difference between the extender position,  $u_e$ , and the human arm position,  $u_h$ , results in contact force,  $f_e$ . The contact force  $f_e$  affects the human arm in the feedback form via  $S_h$ .  $E$ : Flesh Compliance (120lb/rad at DC),  $S_h$ : Arm Sensitivity (0.01 rad/lbf at DC),  $K_f$ : force amplifier gain (.095 V/lbf)

<sup>14</sup> The stability of the system is analyzed by two methods in reference 7. First, the Small Gain Theorem is used to determine a sufficient condition for stability in a completely general, unstructured, nonlinear system. Then, a frequency domain sufficient condition for stability of the linear, time invariant model is determined. The condition for stability is determined using the multivariable Nyquist Criterion, with the "size" of the operators evaluated in terms of singular values. The stability criteria in both cases are expressed in terms of size of  $H$  in comparison with the size of other operators in the loop. It is also shown that the stability condition for linear systems is a sub-class of condition derived by Small Gain Theorem.

Since the experimental extender is a linear one dimensional system, the exact stability can be examined by observing the root locus of the closed loop system. The root locus approaches the imaginary axis as the compensator gain  $K_h$  approaches unity. Thus, the root locus analysis predicts stable operation for  $K_h < 1$  while the system experimentally exhibits stable maneuver for  $K_h < 0.6$ . The stability condition expressed by inequality 12 is a sufficient condition only and it cannot predict instability. Examining inequality 12 leads to a smaller value for  $K_h$  to guarantee the stability, than the one offered by root locus. Although the stability criterion expressed by inequality 12 leads to a more conservative stability condition, it does not depend on the internal structure of the extender and human arm models.

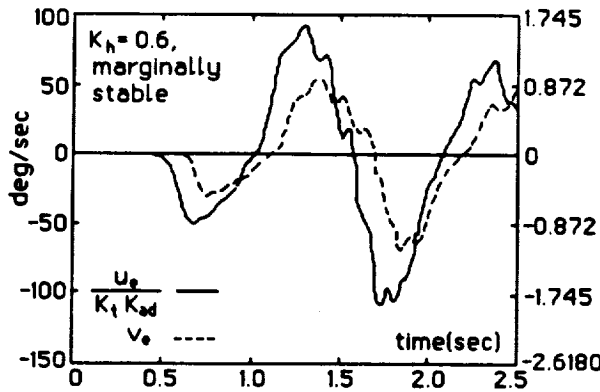


Figure 9: Marginally Stable, ( $K_h = 0.6$ )

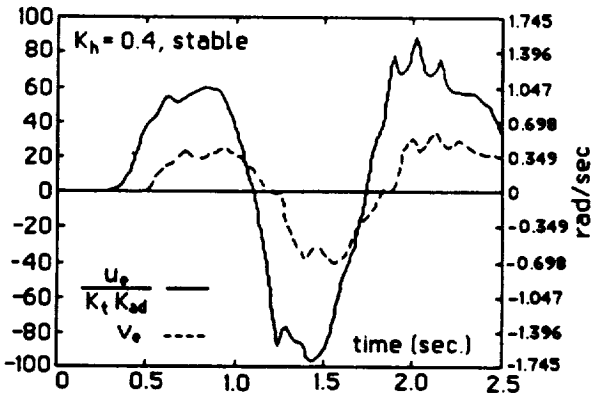


Figure 8: Stable, ( $K_h = 0.4$ )

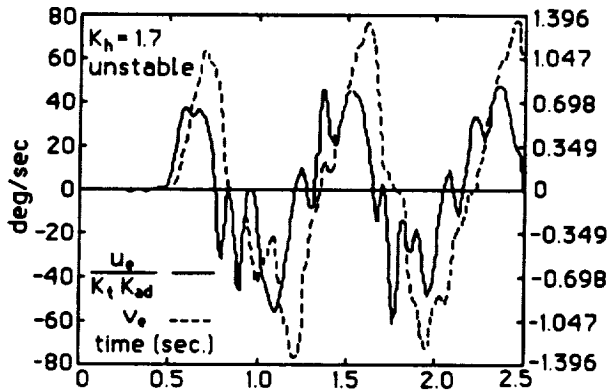


Figure 10: With  $K_h = 1.7$  (unstable case), the human and extender are oscillating  $180^\circ$  out of phase. The extender velocity increases with time.

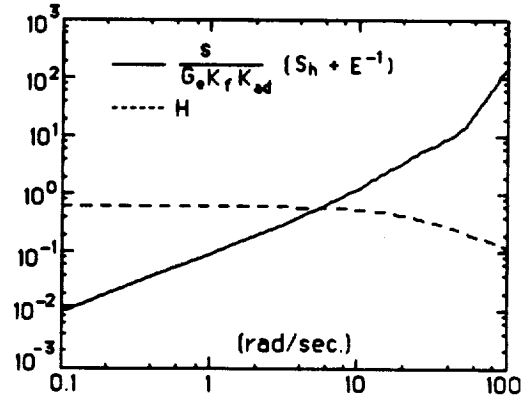


Figure 11: With  $K_h = 1.7$  the system exhibits instability; inequality 12 is not satisfied.

#### 4. Summary and Conclusion

This paper has presented the concept of the extender, which is a manipulator to amplify the strength of a human. Extenders are distinguished from conventional man amplifiers due to their exchange of power and information signals when interacting with the human. The instability of such interaction between the human and extender has been addressed. A hydraulic experimental single degree of freedom extender has been built and tested to verify the control and stability criterion addressed in Part II. A multi degree of freedom extender is being built at the University of Minnesota for research work on the extender constrained maneuvers.

## 5. References

- 1) Anderson, B. J., "An Experimental Study on Physical Human Interaction", M.S. Thesis, March 1988, Mechanical Engineering Dept., University of Minnesota, Minneapolis.
- 2) Battyke, C.K., Nightingale, A. and Whilles, J., Jr., "The Use of Myoelectric Currents in the Operation of Prostheses", Journal of Bone and Joint Surgery, Vol. 37B.
- 3) Bunch, W.H. et al., "Atlas of Orthotics, Biomedical Principles and Application", C.V. Mosby Company, St. Louis, 1985.
- 4) Clark, D.C et al., "Exploratory Investigation of the Man-Amplifier Concept", U.S. Air Force AMRL-TDR-62-89, AD-390070, August 1962.
- 5) Correll, R.W. and Wijnschenck, M.J., "Design and Development of the Case Research Arm Aid", Engineering Design Center Report 4, Case Inst. of Technology, April 1964.
- 6) Doublev, J.A. and Childress, D.S., "Design and Evaluation of a Prosthesis Control System Based on the Concept of Extended Physiological Proprioception", Journal of Rehabilitation Research and Development, Vol. 1, pp.119-131, 1984.
- 7) Foslien, W. K., "On the Stability of Physical Interaction Between Humans and Robotic Systems", M.S. Thesis, November 1987, Mechanical Engineering Dept., University of Minnesota, Minneapolis.
- 8) General Electric Co., "Exoskeleton Prototype Project, Final Report on Phase I", Report S-67-1011, Schenectady, NY, 1966.
- 9) General Electric Co., "Hardiman I Prototype Project, Special Interim Study", Report S-68-1060, Schenectady, NY, 1968.
- 10) Groshaw, P. F., "Hardiman I Arm Test, Hardiman I Prototype", Report S-70-1019, G.E.Co., Schenectady, NY, 1969.
- 11) Hessburg, T.M., "A Theoretical and Experimental Study on Physical Human/Machine Interaction with Constrained Motion", M.S. Thesis, October 1988, Mechanical Engineering Dept., University of Minnesota, Minneapolis.
- 12) Kazerooni, H., Houpt, P.K., Sheridan, T.B., "Fundamentals of Robust Compliant Motion for Manipulators", IEEE Journal of Robotics and Automation, Vol. 2, No. 2, June 1986.
- 13) Kazerooni, H., Houpt, P.K., Sheridan, T.B., "A Design Method for Robust Compliant Motion of Manipulators", IEEE Journal of Robotics and Automation, Vol. 2, No. 2, June 1986.
- 14) Kazerooni, H., Houpt, P. K., "On The Loop Transfer Recovery", Int. J. of Control, Vol. 43, Number 3, March 1986.
- 15) H. Kazerooni, "Stability Criteria for Robot Compliant Maneuvers", In proceeding of the IEEE Int.Conference on Robotics and Automation, Philadelphia, PA, April 1988.
- 16) H. Kazerooni, "Direct-Drive Active Compliant End Effector (Active RCC)", IEEE Journal on Robotics and Automation, Vol. 4, No. 3, June 1988.
- 17) Makinson, B. J., "Research and Development Prototype for Machine Augmentation of Human Strength and Endurance, Hardiman I Project", Report S-71-1056, General Electric Company, Schenectady, NY, 1971.
- 18) Mason, M.T. "Compliance and Force Control for Computer controlled Manipulators", IEEE Trans.on Systems, Man and Cybernetics, Vol. SMC11, No.6, pp.418-432, June 1981.
- 19) Merritt, H. E., "Hydraulic Control Systems", John Wiley & Sons, Inc., 1967.
- 20) Mizen, N. J., "Preliminary Design for the Shoulders and Arms of a Powered, Exoskeletal Structure", Cornell Aeronautical Laboratory Report VO-1692-V-4, 1965.
- 21) Mosher, R. S., "Force Reflecting Electrohydraulic Servomanipulator", Electro-Technology, pp. 138, Dec. 1960.
- 22) Mosher, R. S., "Handyman to Hardiman", SAE Report 670088.
- 23) Rabischong, P. Robotics for the Handicapped, Proc. IFAC Symposium, Columbus, Ohio, May 1982.
- 24) Redford, J.B., "Atlas of Orthotics". Williams and Wilkins, Baltimore, 1986.
- 25) Sheridan, T.B., Ferrell, W.R., "Man-Machine Systems: Information, Control and Decision Models of Human Performance", MIT Press, Cambridge, Massachusetts, 1981.
- 26) Slotine J.-J.E., "The Robust Control of Robot Manipulators", International Journal of Robotics Research, Vol. 4, No. 2, 1985.
- 27) Spong, M.W., Vidyasagar, M., "Robust Nonlinear Control of Robot Manipulators", IEEE Conference on Decision and Control, December 1985.



## **TELEROBOTS 1**





## Trajectory Generation for Space Telerobots

R. Lumia, A. J. Wavering

National Institute of Standards and Technology  
Gaithersburg, MD 20899

### Abstract

The purpose of this paper is to review a variety of trajectory generation techniques which may be applied to space telerobots and to identify problems which need to be addressed in future telerobot motion control systems. As a starting point for the development of motion generation systems for space telerobots, the operation and limitations of traditional path-oriented trajectory generation approaches are discussed. This discussion leads to a description of more advanced techniques which have been demonstrated in research laboratories, and their potential applicability to space telerobots. Examples of this work include systems that incorporate sensory-interactive motion capability and optimal motion planning. Additional considerations which need to be addressed for motion control of a space telerobot are described, such as redundancy resolution and the description and generation of constrained and multi-armed cooperative motions. A task decomposition module for a hierarchical telerobot control system which will serve as a testbed for trajectory generation approaches which address these issues is also discussed briefly.

### 1. Introduction

The Flight Telerobotic Servicer (FTS) has been conceived as a device which will be able to take the place of an extravehicular crew member to perform such tasks as truss assembly, changeout of orbital replacement units (ORUs), electrical and fluid connector coupling and uncoupling, and solar cell array cleaning [27]. To be able to perform these tasks, FTS manipulators must be controlled by a motion generation system which enables the specification, planning, and execution of a wide range of dynamic behaviors. The task decomposition elements of a hierarchical control system which most affect the types of behaviors possible are the servo and trajectory generation software modules. Together, these modules define what types of trajectories, or large dynamic motions, the manipulators can perform (trajectory generation), and how well the manipulator will be able to perform these motions (servo). This paper is concerned primarily with identifying the considerations involved in generating the larger dynamic motions for space telerobots. The preliminary design of a trajectory generation software module for a hierarchical control system is also presented.

### 2. Trajectory Generation Considerations

A natural place to start in the development of trajectory generation software for space telerobots is to examine how the problem is approached for terrestrial manipulators. In this section, important aspects of the description and planning of different types of motions are described. Most of these considerations have emerged from work on earth-based systems, but special implications of operation in the space environment are presented where appropriate.

#### Path Constraints

The most conceptually straightforward, though not the only way to describe a desired manipulator motion is to explicitly define the path which the manipulator should follow through space. Virtually all commercial manipulators use positional paths to specify motions. Paths may usually be specified in either joint space or Cartesian space, and may be represented as equations which are functions of a dimensionless path parameter  $s$ . The path parameter  $s$  indicates the fraction of the path traversed. The motion to perform a given path is created by planning a smooth trajectory function for the path. The trajectory function determines the manipulator position, velocity, and acceleration at any and all times for the duration of the trajectory. Functions for smooth transitions between trajectory segments may also be planned. The trajectory function is evaluated periodically during execution, which results in a sequence of closely-spaced goal points. These goal points are typically sent to individual joint position servo processes, whose task it is to ensure that the

manipulator follow the commanded trajectory as closely as possible. Current trajectory generation techniques which operate in this manner are discussed in detail in [4], [5], and [14]. This type of trajectory planning is sometimes referred to as "constraint satisfaction" [4]. It will be seen that there are other constraints in addition to a position path constraint, which may need to be observed in generating manipulator trajectories.

The path constraint itself is not really an absolute one. Cartesian motion, for example, cannot be performed exactly (for articulated manipulators), since Cartesian goal points must be transformed pointwise into joint coordinates to send to the individual joint servos. If a sufficient number of closely-spaced points are used, a reasonably close approximation to the desired Cartesian path will result. Often, however, it is good enough if a trajectory can be generated which will lie within some position tolerance of the desired path. The tolerance may, in fact, be quite large; particularly for intermediate points in a multi-segment trajectory. If it is specified explicitly, this path tolerance may be used to advantage to simplify the planning of Cartesian straight-line trajectories and to generate more efficient movements. Taylor recognized this, and devised the bounded deviation strategy as an approach to performing Cartesian trajectories [21]. Given a Cartesian straight-line path and an allowable deviation, the bounded deviation strategy will determine the number of points to be traversed in joint space which will keep the trajectory within the stated bounds. Path tolerance can also be seen as an additional freedom in trajectory planning which allows the nominal path to be modified slightly to better achieve the desired objective. The use of path tolerance in creating minimum-time trajectories, for example, is discussed by Suh and Bishop [20].

In planning the trajectory function, conservative limits on manipulator velocity and acceleration are commonly used to prevent planning a motion which cannot be performed due to actuator torque or force limitations. Except in some experimental laboratory systems, the manipulator and payload dynamics are not taken into account during trajectory planning. If the system dynamics are not considered, it is not possible to achieve maximum manipulator performance. This aspect of trajectory planning is discussed in the following section.

### **Manipulator and Payload Dynamics**

The dynamics of the manipulator and payload, along with joint saturation characteristics and environmental interaction forces, determine the achievable motions of the manipulator. For free space motions, dynamics considerations are most crucial when it is desired to extract the maximum performance from a manipulator, as in planning time-optimal motions. In this case the usual conservative limits on path acceleration and velocity are too restrictive. There has been considerable interest recently in developing algorithms that, given a parameterized path specification, will determine the time sequence of torques required to travel the path in minimum time or some other optimal fashion ([18], for example). These algorithms use the manipulator dynamics and actuator torque constraints in computing the optimal trajectory. The dynamics are usually reformulated in terms of the path parameter variable  $s$  and its derivative, rather than joint variables. An optimization technique (dynamic programming, for example) is then used to minimize or maximize the desired performance index subject to the constraints imposed by actuator limitations and manipulator dynamics. An interesting research issue in itself is to identify appropriate objective functions to use for generating trajectories for motions required to perform FTS tasks.

The ping-pong playing robot of Andersson takes manipulator dynamics into account in a somewhat different fashion; the dynamics are used to determine the achievability of postulated motions [2]. That is, a motion is planned to move to a particular state in a certain amount of time, which may or may not be possible. If the planned trajectory is determined to be infeasible, a new motion is postulated, based on some indication of why the former plan could not be performed. This approach has the advantage of reduced computational requirements, since it is only necessary to determine feasibility, not optimality.

The zero-gravity characteristic of the space environment has significant impact on manipulator dynamics, with resulting implications for trajectory planning. Most obvious is that the lack of gravity means that a manipulator does not have to exert actuator torque to support the weight of the manipulator and payload. This implies that there is additional actuator torque available for performing motions, and that there is no explicit payload limitation as with terrestrial manipulators. If the manipulator base is securely attached to a vehicle of sufficient inertia, the limitation is on how quickly objects may be moved around, rather than on the size and mass of the objects. Indeed, several proposed FTS tasks, such as truss assembly and certain ORU changeouts (see [27]), involve handling objects with large masses and/or rotational inertias. This implies that for certain tasks, at least, the payload inertia must be included in the computation of the manipulator dynamics used in trajectory planning and verification.

The lack of gravity also results in an environment in which all objects, including the vehicle or platform

to which the FTS is attached, are free-floating. Since manipulator motions result in forces and torques being transmitted through the base to the supporting object, the supporting object will move in response to manipulator motions unless maneuvering jets or reaction wheels are used to counteract the base forces. This issue is addressed by Vafa and Dubowsky [22]. The implications for manipulator trajectory planning are: 1) the manipulator workspace with respect to the supporting vehicle may be reduced, 2) manipulator motions can cause an undesired change in vehicle attitude, and 3) manipulator motions can produce accelerations which may disrupt micro-gravity experiments on the supporting vehicle. Vafa and Dubowsky approach analysis of the effects of free-floating manipulator configurations through application of a "virtual manipulator," which is a massless arrangement of links which originate from the inertial reference frame located at the system center of mass. The link lengths are determined by the original configuration of the actual manipulator. Once the virtual manipulator has been defined, motions of the actual manipulator are planned by determining the motions of the virtual manipulator which would be required to perform the motion.

### Real-time Sensory Information

Although they may be useful for some free space motions, positional path specifications are not appropriate for all types of manipulator actions. There are many situations where it may not be possible or appropriate to define the desired path of the manipulator *a priori*. For example, when using vision data in real time to perform a trajectory toward a moving object, the path that the manipulator follows in space is determined as the trajectory is being performed. In such cases, it may be more appropriate to simply command a goal state which defines *what* is to be achieved, along with an algorithm specification that defines *how* to achieve it. These types of algorithms perform what is referred to here as *sensory-interactive trajectory generation*. Sensory interactive capabilities will be required for the FTS to perform such tasks as docking with spinning satellites and manipulating objects with a large amount of locational uncertainty (which may be due to shifting during launch or the flexible nature of the object, for example).

There are several possibilities for incorporating vision information into trajectory formation. One may use carefully placed and calibrated cameras to try to locate the target object with respect to the world frame, for example. The calibration requirement of this approach is a major disadvantage, however, since any disturbance of the camera setup will require recalibration. Alternatively, the possibility of using *relative* position differences in camera space to guide motions has been investigated recently [19]. In this approach, the joint position of the manipulator which will achieve the desired camera space relationship of end effector and target object features is repeatedly estimated as the movement is performed. This approach eliminates the reliance on camera calibration and is more robust to camera disturbances.

Another question regarding sensory-interactive motions is how they should be planned and executed. Sensory interaction may be accomplished either by frequent replanning of the trajectory as it is executed or by planning a trajectory function which represents a general profile of the desired motion, with the trajectory details produced dynamically as the trajectory is executed. The first approach has been demonstrated by Andersson for real-time trajectory generation for the ping-pong playing robot [2]. A potential problem with the replanning approach is that there may be a discrepancy between the estimate of what the state of the system will be at the end of planning (which is used as the initial state for the plan) and the actual state at that time. This may result in non-smooth transitions between plans, as the servo module attempts to correct the error. In the approach of determining a general profile, planning may still occur, but it is limited to determining general characteristics of the trajectory implicitly, rather than specifying all aspects explicitly. A set of trajectory parameters is planned which is used to compute the portion of the distance to the goal which should be moved at each cycle. An example of this approach is presented by Myers et al. [12].

### Smoothness

It was mentioned previously that the trajectory functions computed for a path should be "smooth." That is, the resulting manipulator motion should not be jerky. This is particularly true for manipulators in space, since excessive jerkiness can excite structural resonances of both the manipulator arm and the supporting structure, resulting in decreased accuracy and possible instability. Jerky motions also cause accelerated wear of mechanical components. One measure of smoothness is the mean squared magnitude of the rate of change of acceleration (jerk) [9]. Trajectory functions can be planned which give very smooth motion when executed in conjunction with a position servo controller. A position function which is a quintic polynomial of time, for example, results in minimum-jerk motion [9]. However, as stated above, if there is a difference between the estimated initial conditions used to plan the motion and the actual conditions at the beginning of the motion, the corrective servo motions which result will be jerky. An interesting alternative is to use a constant position goal and modulate the gains of the servo module instead of updating the position goals and using

constant gains [7]. If proper gain functions are chosen, the resulting trajectory will always be smooth, and there is no need to preplan position and velocity functions explicitly. Although further analysis and experiments need to be done to assess the usefulness and stability of this technique in actual manipulator applications, it is also interesting in terms of its apparent similarity to some human movements.

### **Closed Kinematic Chains**

Many FTS tasks require contact with another manipulator or other object in the environment, resulting in the formation of a closed kinematic chain which is capable of sustaining internal forces. Some examples of such tasks are truss assembly, tasks where two arms are used to manipulate a single object, and coordinated motions of fingers of a dextrous hand when an object is grasped. The FTS trajectory generation module must be able to plan and execute trajectories for such constrained motions.

One approach for performing constrained actions is to use position-controlled motion along with carefully-engineered passive compliance to guide the motion and prevent excessive interaction forces from being generated [25]. Trajectory planning for constrained motions which use a passive compliance device is based on following a positional path, as described above. This approach has been applied successfully in industrial applications and results in high-speed assembly capabilities, but suffers the drawback of requiring a device which is rather task-specific.

Work has been done to develop appropriate servo control techniques which are more general than a physical device for performing constrained motions. Hybrid position/force control [15], for example, is a control approach which allows the specification of desired position along unconstrained degrees of freedom, and desired forces along those directions which are subject to a position constraint. Given that this type of servo control is available, one has to be able to specify the desired position and force paths and generate the corresponding trajectories which will accomplish a particular task. This has been done primarily for relatively simple tasks and constraint situations, and additional work needs to be done if this approach is to be applicable to more complex tasks. An additional problem is robust on-line determination of the actual constraints the manipulator is subject to during trajectory execution. Some progress has also been made in this area (see [11], for example).

An alternative to the hybrid position/force control approach is to modulate the relationship between manipulator position and force, or the apparent impedance of the manipulator [8]. The first attempts at control of this nature were subsets of generalized impedance control, and include active stiffness control [16] and generalized damper control [26]. An attractive aspect of the impedance control approach is that it represents a step in the direction of approaching contact with the environment as an acceptable, commonplace, and necessary occurrence, rather than as an exceptional circumstance. Trajectory generation for impedance-controlled motions usually consists of planning and executing a nominal position trajectory and controlling the manipulator gains to modulate the impedance, allowing the controlled stiffness, damping, and inertial characteristics to prevent excessive interaction forces during contact. In the case of generalized damper control, a nominal motion direction, rather than path, is used.

Studies of dual-arm cooperative motion controls have also started with identifying useful servo control techniques [17]. Again, standard trajectory generation approaches are typically used with these techniques. The additional problem of synchronization appears when two arms are to perform a coordinated motion. There are two possibilities for two-arm trajectory generation. The first is that each arm has a separate trajectory generation module. The trajectories for each arm are planned independently (although in the same manner), and must be synchronized via some external variable during execution. The other possibility is to have a single trajectory generation module which plans trajectories for both arms at the same time and executes them simultaneously as well. While this approach is straightforward in terms of coordination, there is no spatial decomposition of the task—all of the planning must be performed by a single module, even when the arms are operating independently.

Several important questions regarding constrained motion trajectories remain. What type of static representation for the desired task is most appropriate? How much knowledge of the details of object kinematics and physical properties such as mass, stiffness, and friction coefficient, is needed to plan and execute a trajectory that will accomplish the task? Is there a representation for the task and an approach for performing such actions that simplifies the planning required?

### **Kinematic Redundancy**

Another important aspect of trajectory planning is determining how to most effectively use extra kinematic degrees of freedom which result when the kinematic freedoms of the manipulator exceed those

required by the given task. Extra degrees of freedom may be used to avoid singular manipulator configurations, avoid obstacles, distribute torque requirements more evenly among the joints, and achieve similar motion subgoals, while still allowing the end effector to follow a prescribed path. For space applications, an additional useful possibility might be to use redundant degrees of freedom, or self motion, to control base reaction forces and disturbances to the supporting vehicle. The manipulators for the FTS will have redundant degree(s) of freedom with respect to a six degree-of-freedom positioning task, and kinematically-redundant commercial manipulators have recently been introduced. Redundancy also occurs when a six degree-of-freedom arm is equipped with a dextrous multi-fingered hand. Some means of distributing the desired motion between the hand and the arm is required.

One possibility for resolving redundancy is to use global optimization techniques to determine the joint positions or torques for an entire trajectory as in [13]. Nakamura's approach makes use of Pontryagin's Maximum Principle to determine how the extra degree(s) of freedom may best be used over an entire trajectory to globally optimize a specified performance index. Although this approach results in global optimization, it is extremely computationally intensive and therefore useful primarily for off-line computation.

An alternative to resolving redundancy globally over an entire trajectory is to perform local kinematic inversion for points along a Cartesian trajectory as the motion is being executed. In this case, redundancy resolution takes place during the execution, rather than the planning, of a trajectory. Although they do not result in global optimization of any objective function, there are local redundancy resolution techniques which may be performed sufficiently fast as to be used on-line during trajectory execution. Reference [3] presents a survey of a number of these techniques.

### Operator Interaction

Not only must the control system be able to accomplish tasks autonomously, but it must also be able to accept operator input at all levels to allow teleoperation and execution of operator commands. Most of what is usually thought of as teleoperation—that is, direct manipulator control by an operator—is performed by the servo module, rather the trajectory generation module. For teleoperation, the operator performs trajectory generation directly by manipulating a master arm or other control device. Interaction with the trajectory generation module consists primarily of entering static motion commands of the same type which are sent to the trajectory generation module when the system is operating autonomously. This gives the operator the capability of indicating a desired path or goal state, and allowing the trajectory generation module to plan and execute the desired motion. In addition to this type of interaction, it is also desirable to give the operator an overriding control of the manipulator velocity, so that the operator can slow down or stop the manipulator at any time during trajectory execution.

### 3. Trajectory Generation Software Module Design

The aspects of trajectory generation outlined above have been considered in the design of a trajectory generation module for a hierarchical manipulator control system. The overall framework of such a system for autonomous and teleoperated telerobot control is described in [1]. The trajectory generation module is part of the *task decomposition* hierarchy, which subdivides high-level tasks into simpler and simpler subtasks. There are also separate hierarchies which perform *sensory processing* and *world modeling* functions. World and manipulator state information and communication interfaces are contained in a *global data system*, available to all processes. The reader is referred to [1] for additional details on these parts of the system. The decomposition performed by the trajectory generation module is to generate a time sequence of closely-spaced manipulator goal states from a static description of the desired motion. As such, it generates primitive trajectories, and is called the Primitive (Prim) task decomposition module.

During autonomous operation, Prim receives commands from the Elemental Move (E-move) level of the task decomposition hierarchy, which performs such functions as grasp planning and obstacle-free path planning. The Prim module can also accept commands from the Operator Control. The output commands of the E-move level are time-independent descriptions of motions, for example static position or position and force paths. In addition to these types of commands, E-move can also simply specify a set of termination conditions, or goal states, along with an algorithm specification which determines the strategy to be used achieve them. This type of specification is useful with sensory interactive trajectory algorithms, where the exact path is determined as the motion is performed, based on sensed data. Commands entered by the operator through the Operator Control contain the same information and have the same format as those which come from E-move.

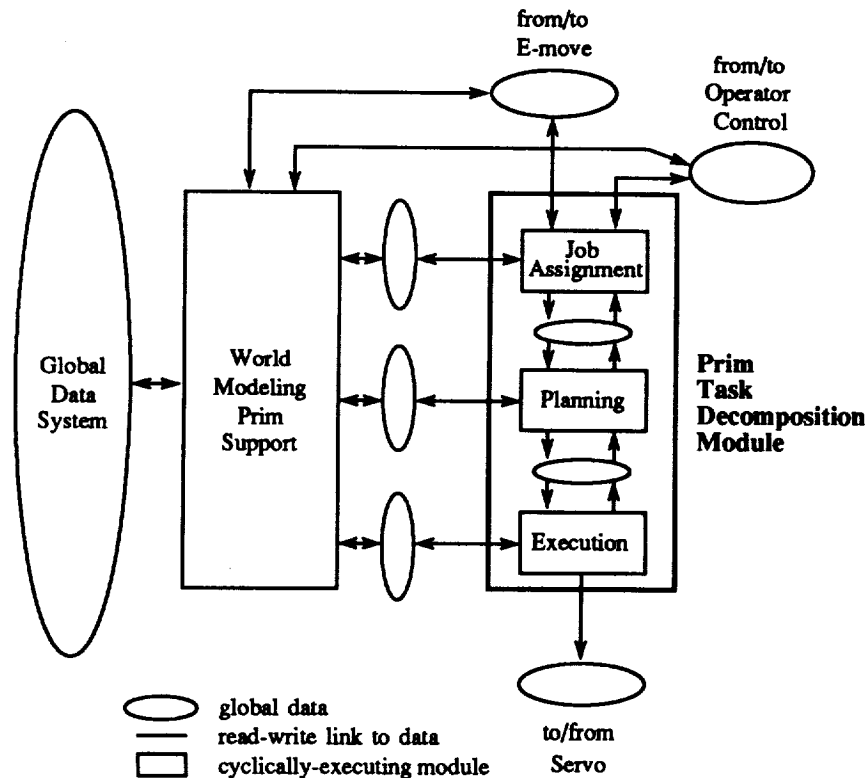


Figure 1. Prim Task Decomposition Structure.

Prim generates the time sequence of attractor sets needed to produce a dynamic trajectory from the E-move or Operator Control command, and sends these as commands to the Servo level of the task decomposition hierarchy. The Servo level, the lowest level in the task decomposition hierarchy, controls the behavior of the manipulator in performing small motions between closely-spaced goals. The function and interfaces of a Servo level for manipulators which accommodates a broad spectrum of published control algorithms is described in [6]. In addition to determining position, velocity, acceleration, and/or force trajectories to be commanded to Servo, Prim also has the task of determining appropriate manipulator impedance, stiffness, damping, and inertial characteristics which are controlled by adjusting the servo loop gains commanded to Servo.

### Structure

As illustrated in Figure 1, the Primitive task decomposition module is composed of three concurrent, cyclically-executing processes; the Job Assignment module, the Planning module, and the Execution module. These processes execute independently of one another and have different cycle times, with the Execution module typically operating at a much higher rate than the Job Assignment and Planning modules. Each of these submodules continually repeats a cycle which consists of reading inputs, performing computations, and writing outputs. This type of operation prevents the entire system from locking up if a single process hangs during an attempt to compute or communicate. Such freedom from system lock-up is an essential feature for safe and reliable manipulator control. The functions of the three submodules are discussed below.

The Job Assignment module coordinates transitions between autonomous and operator control through management of the input command queue. The input commands to Prim are queued so that future commands will be available to the Planning module in order to plan transitions between motion segments. When the operator wishes to take control at the Prim level, he or she may do so by editing the queue to insert and delete commands.

The Prim Planning module handles the generation of a plan for a dynamic trajectory (for example, time

functions of manipulator position, velocity, acceleration, and/or force). It is important to realize, however, that the terms "trajectory planning" and "trajectory generation algorithm" as used here do not always refer to methods of preplanning the exact position, velocity, and acceleration of the manipulator at every instant during the motion. Instead, the Planning module may only determine functions or parameters which determine what the general profile should look like for the motion, and the exact path the robot takes during execution is determined by sensory or other external inputs. For planning all types of motions, the Prim Planning module looks ahead by an amount of time which depends on several factors, including the length of the motion itself, the trajectory generation algorithm used, and the nature of the objective function to be optimized during the motion. The Prim planning horizon will typically be on the order of 100 ms, although it may be as much as one to a several seconds.

In addition to planning a suitable trajectory, the Planning module must also select an appropriate servo algorithm and servo loop gains. The position, force, and torque servo loop gains determine the behavior of the manipulator in response to new position and force goals, and to external disturbances. The Prim Planning module may determine the apparent manipulator impedance by adjusting these gains. A constant set of gains that works reasonably well for a variety of manipulator configurations and payloads might be used for free space moves. Such gains result in compromised performance, however, since the manipulator dynamics are configuration-dependent. Alternatively, gains may be varied as a function of the manipulator state by the Execution module during trajectory execution.

Another task performed by the Planning module is redundancy resolution when a global optimization technique is used to transform a six degree-of-freedom path specification into a seven or more degree-of-freedom joint trajectory. The Planning module also determines the intervals of time for which the position and force trajectory functions should be evaluated.

The Prim Execution module has two primary functions. First, it must evaluate the position, velocity, acceleration, jerk, force, and time derivative of force functions of time for the intervals specified by the Planning module. This results in the point attractor vectors which are sent as commands to the Servo level. In addition, the Execution module is responsible for monitoring position, velocity, force, and other sensor states or world model conditions for achievement of the termination conditions. The Execution module must also monitor the commands it sends to Servo to make sure they are achievable. This includes making sure that excessive joint velocities are commanded, even if the manipulator is near a singular configuration. Also, the resolution of redundancy, if it is performed by kinematic criteria, is performed by the Execution module. Furthermore, the Execution module incorporates operator velocity control and single-step interactions. The Execution module also should calculate the estimated termination time for motions when this is possible.

Information about the state of the manipulator and the world is needed to perform planning and execution tasks. This information is provided to the task decomposition module via the world modeling support module shown in Figure 1. This world modeling support module contains processes which access data stored in the global data system by the sensory processing side of the control hierarchy, and perform model-based computations (such as manipulator kinematics and dynamics). The Prim world modeling support module may access information which has been processed by any level of the sensory processing hierarchy. Note that this implies there may not be a direct correspondence between sensory processing levels and task decomposition levels.

## **Interfaces**

An attempt has been made to identify the types of information which should be included in the Prim command and status interfaces to allow the implementation of trajectory generation algorithms which take into account the considerations presented in Section 2. The Prim input command and output status interface information is given in Table 1. The command specification consists of an algorithm and a set of parameters. The parameters which have been included represent commonly-used means of describing manipulator motions in a time-independent manner, and expressing what factors are important in transforming the command into a dynamic movement. The interface parameters are discussed in detail in [23,24].

## **Implementation**

An implementation of the trajectory generation module described above is currently being developed in Ada. Clearly, the development of such a module with all of the desired capabilities is quite a formidable task. The approach which has been taken is to implement a skeletal module which contains the desired concurrent functional submodules (Job Assignment, Planning, and Execution), and interface variables. A small number of simple trajectory generation algorithms are being implemented initially, along with basic

Table 1. E-move to Primitive Interface Elements.

<u>Primitive Input Command Elements</u>	<u>Primitive Output Status Elements</u>
Command number	Job Assignment status
Prim algorithm	Planning command number
Coordinate system	Planning status
Position command description	Execution command number
Force command description	Execution status basis
Held object	Estimated termination time
Destination object	
Termination condition(s)	
Redundancy resolution specification	
Priority	
Objective function	

world modeling functions. Building upon this foundation, additional algorithms and capabilities will be added. The mapping of the system architecture into computing hardware for this implementation is discussed in [10].

#### 4. Conclusions

The capabilities desired of the FTS place rather severe demands on the control system trajectory generation module. Although significant progress has been made in developing advanced techniques for trajectory generation, continued work is needed in the areas of task representation, constrained motion generation, redundant manipulator control, coordination of multiple arms with dextrous end effectors, and vision servoing. Further investigations addressing the use of manipulator dynamics in trajectory generation are also needed. Although not discussed in this paper, advances must also occur in the techniques used in the sensory processing and world modeling hierarchies to provide the information needed for advanced trajectory generation.

This paper has not addressed the hardware and software requirements imposed if all of the considerations discussed are to be included. The requirements are quite formidable, however, and a logical approach is to develop first a basic structure which allows for the implementation of many different algorithms of varying complexities. One may then start with straightforward, proven algorithms and proceed to add capabilities as improvements in algorithms and computing hardware come about. This is the approach which has been taken at NIST for motion control system development.

#### 5. References

- [1] Albus, J. S., McCain, H. G., Lumia, R., NASA/NBS Standard Reference Model for Telerobot Control System Architecture (NASREM), NASA Document SS-GSFC-0027, December 4, 1986.
- [2] Andersson, R. L., "Agressive Trajectory Generator for a Robot Ping-Pong Player," Proc. IEEE Conf. Robotics and Automation, Philadelphia, PA, April 1988.
- [3] Baillieul, J., et al., "Kinematically Redundant Manipulators," Proc. NASA/JPL Workshop on Space Telerobotics, Pasadena, CA, January 1987.
- [4] Brady, M., et al., eds., Robot Motion: Planning and Control, MIT Press, Cambridge, Mass., 1982.
- [5] Craig, J. J., Introduction to Robotics: Mechanics and Control, Addison-Wesley, Reading, Mass., 1986.
- [6] Fiala, J. C., "Manipulator Servo Level Task Decomposition," NIST Technical Note 1255, NIST, Gaithersburg, MD, October 1988.
- [7] Fiala, J. C., "Generation of Smooth Trajectories without Planning," submitted to 1989 IEEE Int. Conf. on Robotics and Automation.
- [8] Hogan, N., "Impedance Control: An Approach to Manipulation," Journal of Dyn. Sys., Meas., and Control, March 1985.



- [9] Hogan, N., "An Organizing Principle for a Class of Voluntary Movements," Journal of Neuroscience, Vol. 4, No. 11, November 1984.
- [10] Lumia, R., Fiala, J. C., "The FTS: From Functional Architecture to Computer Architecture," submitted to 1989 NASA/JPL Workshop on Space Telerobotics.
- [11] Merlet, J-P., "C-surface applied to the design of an Hybrid Force-Position Robot Controller," Proc. IEEE Int. Conf. on Robotics and Automation, Raleigh, NC, March 1987.
- [12] Myers, D. R., Leake, S. A., Juberts, M., "Control System Architecture for Telemanipulator Operation," in Recent Trends in Robotics: Modeling, Control, and Education, M. Jamshidi, J. Y. S. Luh, M. Shahinpoor, eds., North-Holland, New York, NY, 1986.
- [13] Nakamura, Y., Hanafusa, H., Yoshikawa, T., "Task Priority Based Redundancy Control of Robot Manipulators," Inter. Journal of Robotics Research, Vol. 6, No. 2, Summer 1987.
- [14] Paul, R. P., Robot Manipulators: Mathematics, Programming, and Control, MIT Press, Cambridge, MA, 1981.
- [15] Raibert, M. H., Craig, J. J., "Hybrid Position/Force Control of Manipulators," Transactions of the ASME, Vol. 102, June 1981.
- [16] Salisbury, J. K., "Active Stiffness Control of a Manipulator in Cartesian Coordinates," Proc. 19th IEEE Conf. Decision and Control, Albuquerque, NM, December 1980.
- [17] Seraji, H., "Adaptive Hybrid Control of Manipulators," Proc. NASA/JPL Workshop on Space Telerobotics, Pasadena, CA, January 1987.
- [18] Shin, K. G., McKay, N. D., "A Dynamic Programming Approach to Trajectory Planning of Robotic Manipulators," IEEE Trans. on Automatic Control, Vol AC-30, No. 6, June, 1986.
- [19] Skaar, S. B., Brockman, W. H., Hanson, R., "Camera Space Manipulation," Inter. Journal of Robotics Research, Vol. 6, No. 4, 1987.
- [20] Suh, S. H., Bishop, A. B., "The Tube Concept and its Application to the Obstacle-Avoiding Minimum-Time Trajectory Planning Problem," Proc. IEEE Int. Conf. on Sys., Man, and Cybernetics, Alexandria, VA, October 1987.
- [21] Taylor, R. H., "Planning and Execution of Straight Line Manipulator Trajectories," IBM J. Res. Develop., Vol. 23, No. 4, 1979.
- [22] Vafa, Z., Dubowsky, S., "On the Dynamics of Manipulators in Space Using the Virtual Manipulator Approach," Proc. IEEE Int. Conf. on Robotics and Automation, Raleigh, NC, March 1987.
- [23] Wavering, A. J., "Manipulator Primitive Level Task Decomposition," NIST Technical Note 1256, NIST, Gaithersburg, MD, October 1988.
- [24] Wavering, A. J., Lumia, R., "Task Decomposition Module for Telerobot Trajectory Generation," Proc. SPIE Space Station Automation IV Conference, Cambridge, MA, November 1988.
- [25] Whitney, D. E., Nevins, J. L., "What is the RCC and What Can It Do?," Proc. 9th Int. Symp. on Industrial Robots, March 1979.
- [26] Whitney, D. E., "Force Feedback of Manipulator Fine Motions," Journal of Dynamic Sys., Meas., and Control, December 1982.
- [27] Zimmerman, W., Myers, J., Ruth, D., "Task Ranking for the Telerobot Demonstration Final Report," JPL Contract 957908, July 1988.



# On the Simulation of Space Based Manipulators with Contact <sup>1</sup>

Michael W. Walker and Joseph Dionise

Robotics Research Laboratory  
Electrical Engineering and Computer Science Department  
The University of Michigan  
Ann Arbor, Michigan 48109

## Abstract

This paper presents an efficient method of simulating the motion of space based manipulators. Since the manipulators will come into contact with different objects in their environment while carrying out different task, an important part of the simulation is the modeling of those contacts. An inverse dynamics controller is used to control a two armed manipulator whose task is to grasp an object floating in space. Simulation results are presented and an evaluation is made the performance of the controller.

## 1 Introduction

Robotic manipulators carried by future spacecraft are expected to perform many important tasks in space. This paper presents a methodology for the simulation and control of these robots. The main idea is the need to include not only the dynamics of the robot in the simulation, but also the dynamics of the objects in which the robot comes into contact. The same is true in the design of the controller. If the robot is to be used in grappling a satellite, then the controller and the simulation must include a model of the satellite dynamics.

The model we use is called the dynamic world model as it includes the dynamics of the objects in the robots environment and the dynamics of the robot itself. The next section presents the formulation of this world model. Included in the model is the dynamics of contact. A recently reported distance algorithm is used to detect contact [4].

The next section deals with the control of the robot. Feedback functions can be utilized by the controller either at the joint level or at the Cartesian level. An example is provided for an implementation of inverse dynamics control.

The next section presents an example simulation. In this simulation, two PUMA 560 manipulators are mounted on a single base. The task is to grapple an object floating in space. The inverse dynamics control algorithm which has been developed for terrestrial-based robots is used for control. Compliance is obtained by using a relatively small position error gains in the controller.

The final section concludes the paper with a discussion of the merits and limitations of the method.

## 2 Dynamic World Model

This section presents the world model used by the simulation. First, the data structure which is used to store the model is presented. Next, the kinematic modeling of the system is presented. The systems dynamic equations of motion are then presented followed by the dynamics of contact.

<sup>1</sup>This work was supported in part by a grant from the NASA sponsored Center for Autonomous and Man-Controlled Robotic and Sensing Systems, CAMRSS, at ERIM, Ann Arbor, MI



## 2.1 The Data Structure

In general, the number of rigid links in the system is  $m+1$ . For the example in Figure 1(a),  $m+1=8$ . The inertial link is a imaginary link fixed with respect to an inertial reference frame and numbered 0. All other links are numbered in an arbitrary order from 1 to  $m$ . It is apparent that a graph data structure could be used to store the information. Each record or item in the data structure would be associated with a particular link and all of the information needed about that link would be stored in the associated data record. The problem with using this type of data structure is that it does not lead to a particularly simple or efficient algorithm for the controller. A much better data structure is a binary tree data structure. The remainder of this section develops the method of obtaining this type of data structure.

An example of a link with four joints is illustrated in Figure 2. The joint connecting the immediate predecessor to the link is assigned the same number as the link. One of the other three joints is used to locate the link coordinates. This determines the D-H kinematic parameters  $a_i$ ,  $\alpha_i$ ,  $d_i$ , and  $\theta_i$  as illustrated in the figure. To locate the remaining two joints with respect to link  $i$  coordinates we insert fictitious links,  $j$  and  $k$ , called connector links. First is link  $j$  which is located relative to link  $i$  coordinates. The D-H kinematic parameters  $[1]$ ,  $a_j$ ,  $\alpha_j$ ,  $d_j$ , and  $\theta_j$  are used to locate this link  $j$  coordinates with respect to link  $i$  coordinates. Next is link  $k$  which is located relative to link  $j$  coordinates. Again the D-H kinematic parameters,  $a_k$ ,  $\alpha_k$ ,  $d_k$ , and  $\theta_k$  are used to locate this link  $k$  coordinates with respect to link  $j$  coordinates. Note that all of the kinematic parameters,  $a$ ,  $\alpha$ ,  $d$ , and  $\theta$ , associated with connector links are constant and that for both the connector and successor links the position and orientation of the descendant with respect to the current link

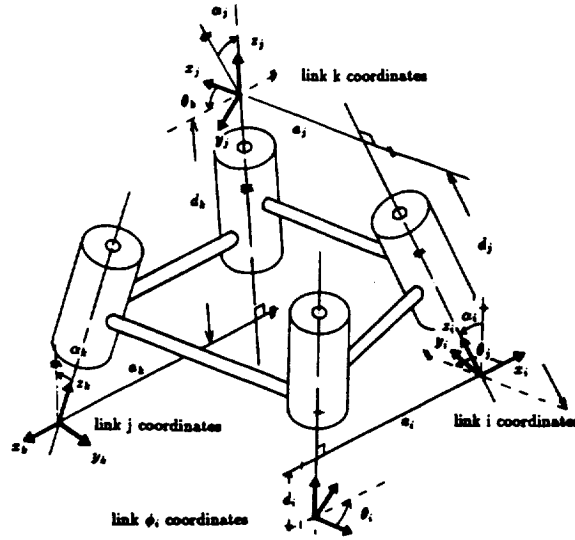


Figure 2: An Example Link, Numbered  $i$

is described using homogeneous transforms parameterized by the D-H kinematic parameters  $a$ ,  $\alpha$ ,  $d$  and  $\theta$ . The real links are called successor links to distinguish them from the imaginary connector links.

Therefore, each link can only have two immediate descendants. One would be a connector link and the other would be a successor link. Only one of each type is allowed. Hence, with the introduction of the connector links we have converted a general tree data structure into a binary tree data structure.

For a given link, the first joint encountered when moving in the direction of the inertial link is numbered the same as the given link. Thus, all of the joints except those where the kinematic loops have been broken have been assigned a number. There are  $m$  of these. Assuming there are  $r$  independent loops the remaining  $r$  joints located at the points where the kinematic loops have been broken are numbered from  $m + 1$  to  $m + r$  making a total of  $m + r$  joints. Finally, a fictitious successor link called a terminating link is associated with these joints and is assigned the same number as the associated joint. The purpose of the terminating link is to allocate a item in the data structure to store all of the information concerning the associated joints, for example, their position and the viscous friction coefficients. The D-H kinematic parameters for the terminating links are chosen so that the loop closure equations can be written in terms of homogeneous transforms. This will be described in more detail in the section concerning the constraint equations.

If link  $i$  is a successor link then the associated joint is either translational or rotational. In either case the joint position is denoted by  $q_i$ . If the joint is translational then  $q_i = d_i$ . If the joint is rotational then  $q_i = \theta_i$ .

Finally, connector links are numbered starting at  $m + r + 1$  on up to the total number of links, both imaginary and real, contained in the system.

The resulting data structure for the example system is illustrated in Figure 1(b). The convention we use is that the items in the data structure associated with connector links are connected to their predecessor with small solid circles and items associated with successor links are connected to their predecessor with small white circles.

## 2.2 System Kinematics

From Figure 1(b) one can see that the position of each link can be determined with respect to the inertial coordinates by starting at the inertial link and successively computing each link position while moving out from the inertial link. Let  $\phi_i$  denote the number of the immediate predecessor of link  $i$ . If the homogeneous transform of link  $\phi_i$  coordinates with respect to the inertial link 0 coordinates,  $T_0^{\phi_i}$ , is known then the homogeneous transform of link  $i$  coordinates with respect to the inertial link is  $T_0^i$  and can be computing using the following equation.

$$T_0^i = T_0^{\phi_i} T_{\phi_i}^i$$

where  $T_{\phi_i}^i$  is a function of the  $a_i$ ,  $d_i$ ,  $\alpha_i$ ,  $\theta_i$ . Note that the same equation applies if link  $i$  is a connector link or a successor link.

This paper uses the spatial notation by Featherstone, [3,5]. With this notation transformation matrices are  $6 \times 6$  matrices. The spatial transformation matrix from link  $i$  to link  $\phi_i$  is:

$$X_{\phi_i}^i = \begin{bmatrix} A_{\phi_i}^i & O \\ K(p_{\phi_i}^i)A_{\phi_i}^i & A_{\phi_i}^i \end{bmatrix}$$

where  $O$  is the  $3 \times 3$  null matrix,  $A_{\phi_i}^i$  and  $p_{\phi_i}^i$  are the upper left  $3 \times 3$  submatrix and upper right  $3 \times 1$  submatrix of the homogeneous transformation matrix  $T_{\phi_i}^i$ , respectively. The  $3 \times 3$  matrix  $K()$  is a skew symmetric matrix such that  $K(a)b = a \times b$  for any  $3 \times 1$  vectors  $a$  and  $b$ .

As with homogeneous transforms, the spatial transformation from link  $i$  coordinates to link 0 coordinates can be computed given that of its immediate predecessor by multiplying the transforms together.

$$X_0^i = X_0^{\phi_i} X_{\phi_i}^i \quad (1)$$

The spatial velocity and acceleration of link  $i$  can be determined given those of its immediate predecessor,  $\phi_i$ .

$$v_i = \begin{cases} v_{\phi_i} + s_i \dot{q}_i & \text{if link } i \text{ is a successor} \\ v_{\phi_i} & \text{if link } i \text{ is a connector} \end{cases} \quad (2)$$

$$\dot{v}_i = \begin{cases} \dot{v}_{\phi_i} + s_i \ddot{q}_i + v_i \times s_i \dot{q}_i & \text{if link } i \text{ is a successor} \\ \dot{v}_{\phi_i} & \text{if link } i \text{ is a connector} \end{cases} \quad (3)$$

where  $s_i$  is the third column of  $X_0^{\phi_i}$  if joint  $i$  is rotational or the sixth column of  $X_0^{\phi_i}$  if joint  $i$  is translational.

Thus, one starts at the root of the binary tree data structure and works out along the branches using the above equations to successively compute the spatial transformation matrix, velocity and the acceleration of each link.

### 2.3 Kinematic Constraint Equations

In general, the system is graph structured with  $n$  degrees of freedom. For this reason, we partition the set of joints into two mutually exclusive sets. There are  $n$  joints in the first set. They are called primary joints since their positions are independent of any other joint positions in the system. These joint positions are combined into a single  $n \times 1$  vector called,  $Q$ . The joints in the second set are called secondary joints, as their positions are strictly functions of the primary joint positions. Thus, the positions of all the joints in the system are functions of the primary joint positions. We can write this fact in the form of the following constraint equation.

$$q = U(Q) \quad (4)$$

Note that for each  $Q_i$  there is a  $q_j$  such that  $Q_i \equiv q_j$ . Given  $U(Q)$ ,  $\dot{Q}$ , and  $\ddot{Q}$ , the velocity and acceleration of all the joints can be determined.

$$\dot{q} = E(Q)\dot{Q} \quad (5)$$

$$\ddot{q} = E(Q)\ddot{Q} + \dot{E}(Q)\dot{Q} \quad (6)$$

where

$$E(Q) = \frac{\partial U(Q)}{\partial Q} \quad (7)$$

For a given system these equations are usually fairly simple. Often the matrix  $E(Q)$  is constant. For example, it is simply the identity matrix for a tree structured system. However, for some systems these equations can become complex. For these systems we have found that the  $\epsilon$ -algebra is very convenient for evaluating the time derivatives of the joint positions [9]. Using this algebra one only has to program to solution to Equation 4, change the order of the algebra and automatically obtain the solution to equations 5 and 6.

The equations of constraint are obtained from the loop closure equations. For the system shown in Figure 1(a) there are two independent loops. The two loop closure equations are:

$$T_0^1(q_1)T_1^{13} = T_0^{10}T_{10}^{11}T_{11}^4(q_4)T_4^5(q_5)T_5^8(q_8) \quad (8)$$

and

$$T_0^1(q_1)T_1^2(q_2)T_2^{12} = T_0^{10}T_{10}^6(q_6)T_6^7(q_7)T_7^9(q_9) \quad (9)$$

These two matrix equations consist of a total of 32 scalar equations of which only six are independent. Thus, we can determine  $q_1$ ,  $q_2$ ,  $q_4$ ,  $q_6$ ,  $q_8$ , and  $q_9$  as a function of the joint positions  $q_5$  and  $q_7$ . Using the above notation,  $q_5 = Q_1$ ,  $q_7 = Q_2$  and  $q_3 = Q_3$ . These three equations along with the six obtained from equations 8 and 9 give us the function  $U(Q)$  defined above.

## 2.4 System Dynamics

The equations of motion of a system can be written in the following form:

$$\tau = H(Q)\ddot{Q} + C(Q, \dot{Q})\dot{Q} + G_e(Q)$$

where

- $\tau$  =  $n \times 1$  vector of active forces
- $H(Q)$  =  $n \times n$  pseudo moment of inertia matrix
- $C(Q, \dot{Q})\dot{Q}$  =  $n \times 1$  vector of friction, centrifugal and Coriolis terms
- $G_e(Q)$  =  $n \times 1$  vector of external force components

In the simulation program the state variables are the position and velocity of the primary variables,  $Q$ . To simulate the system one determines the acceleration of these variables from the above equation:

$$\ddot{Q} = H(Q)^{-1}(\tau - C(Q, \dot{Q})\dot{Q} - G_e(Q))$$

We use the first method presented in [10] to compute the  $H(Q)$  matrix. Although this method was originally presented for serial link manipulators, the approach is still applicable if one uses an inverse dynamics algorithm for systems with closed kinematic loops [6,7,8].

The inverse dynamics algorithm we use is a three step process [8].

1. Given the desired positions, velocities, and accelerations of the independent variables,  $Q$ , calculate the corresponding positions, velocities, and accelerations of the joints,  $q$ , using equations, 4 through 6.
2. For each link, determine  $\rho_j$ , the sum of the friction force and the projection of the sum of the inertial forces on the axis of motion of joint  $j$ .
3. Determine the active forces,  $\tau$  using the constraint equations and the  $\rho_j$  computed in step 2.

$$\tau = E(Q)^T \rho$$

where  $\tau$  is an  $n \times 1$  vector of the  $\tau_j$  and  $\rho$  is an  $(m+r) \times 1$  vector of the  $\rho_j$ .

Step 2 is exactly the same procedure which would be used if the system was a tree structure system. In this case the  $\rho_j$  would be identical to the active forces. The only difference is in the first step, wherein joint positions, velocities, and accelerations are determined which are consistent with the constraint equations, and the last step, wherein the constraint equations are again used in determining the active forces.

Step 2 can be implemented in three steps. First, the position, velocity and acceleration of each link is computed using equations 1 through 3. This is done by starting at the inertial link and working out along the branches of the tree. Next, for each link  $j$  we determine  $f_j$ , the sum of the external forces and inertial forces of link  $j$  and all of its descendants. This is done by starting at the tips of the branches of the tree and working back toward the inertial link of the tree using the following equations. Let  $F_j$  denote the inertial force of link  $j$ . Then:

$$F_j = \dot{M}_j = I_j \dot{v}_j + v_j \times I_j v_j \quad (10)$$

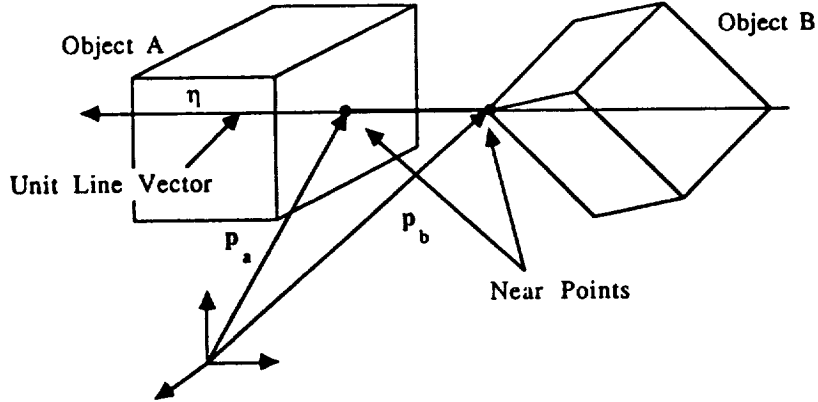


Figure 3: Illustration of distance between two convex polytopes

where  $M_j = I_j v_j$  is the spatial momentum vector for the  $j$ -th link and  $I_j$  is the spatial moment of inertia matrix referred to inertial coordinates:

$$I_j = X_0^j L_j X_0^{j'}$$

In this equation the matrix  $L_j$  is the spatial moment of inertia matrix referred in link  $j$  coordinates:

$$L_j = \begin{bmatrix} -m_j K(p_j) & m_j I \\ L_j & m_j K(p_j) \end{bmatrix}$$

where  $m_j$  is the mass of link  $j$ ,  $p_j$  is the location of the center of mass, and  $L_j$  is the moment of inertia matrix. Both  $p_j$  and  $L_j$  are defined with respect to link  $j$  coordinates. The  $3 \times 3$  skew symmetric matrix  $K(p_j)$  is a function of the vector  $p_j$  such that for any vector  $a$ ,  $K(p_j)a = p_j \times a$ .

For connector and terminating links the  $F_j$  are zero since they have no mass. Let  $j$  be the index of the current link, and let  $k$  and  $l$  be the index of its successor and connector links. The vector  $f_j$  is used to denote the sum of the inertial and external forces for link  $j$  and all of its descendants.

$$f_j = F_j + F_j^e + f_k + f_l \quad (11)$$

The last part of step 2 is to determine  $\rho_j$ , the force due to friction,  $\eta_j$ , plus the projection of  $f_j$  onto the axis of motion of joint  $j$ .

$$\rho_j = s_j' f_j + \eta_j \quad (12)$$

where  $\eta_j = d_j \dot{q}_j$  is the component of joint force due to viscous friction.

## 2.5 Contact Dynamics

To simulate contact we need a geometric model of the links in the system and a method of computing the distance between these objects. We use unions of convex polytopes to model the geometry of each link. An example of two objects is illustrated in figure 3. Polytopes were used as a model since the shape of links can be closely approximated by using a suitable number of vertices in the polytopes and a very efficient distance algorithm exist for computing the distance between convex polytopes, [4]. Given the position of the two polytopes the algorithm returns the distance,  $d$ , between them as well as the location of the point in each polytope which is nearest the other polytope,  $p_a$  and  $p_b$ . In addition, the unit line vector,  $\eta$ , is returned which passes through the near points of the two objects.



The distance between objects is calculated after each integration step in the simulation. The integration algorithm is a variable step size algorithm. When collision has not occurred, the integration step size is limited to ensure that each joint position changes by no more than a specified amount. As objects near each other, the step size is also limited so the objects will just touch at the next integration step. We say that two objects are just touching when the distance between them is equal to a small constant,  $\epsilon$ . An estimate of the time when the objects will collide is given by the following equation.

$$\text{collision time} = \frac{\epsilon - d}{\dot{d}}$$

If the collision time is greater than zero, the integration step size is selected to be less than or equal to this value. If the collision time is negative, then the objects are moving apart from each other and this collision time is ignored.

The value of  $\dot{d}$  is easily computed from the unit line vector,  $\eta$ , returned from the distance algorithm and the spatial velocities of the two objects,  $v_a$  and  $v_b$ , which are obtained from the simulation algorithm.

$$\dot{d} = \eta'(v_a - v_b)$$

The dynamics of contact is modeled like a spring. When the distance between the objects is less than  $\epsilon$  a force is applied to object  $A$  in the direction of  $\eta$  with magnitude,  $f_c$ , given by:

$$f_c = K_p \frac{\epsilon - d}{d} - K_d \dot{d}$$

where  $K_p$  and  $K_d$  are positive constants. Thus, the magnitude approaches infinity as the actual distance,  $d$ , approaches zero. The objective of this is to ensure that the objects never intersect. The derivative term is added to provide damping. The spatial force,  $f_e$  exerted on object  $A$  is given by:

$$f_e = f_c \eta$$

Thus, we are modeling hard contacts with no sliding friction. The spatial force exerted on object  $B$  is the negative of this value. These forces are included into the dynamic simulation as external forces exerted on the links as described in the above section on dynamics.

### 3 Control

Feedback signals used by the controllers come in two forms: spatial feedback vectors,  $\hat{F}_j$ , which are associated with the links of the manipulator and scalar feedback functions,  $\hat{\eta}_j$ , which are associated with the joints of the manipulator. These can be any functions of the desired trajectory and the actual trajectory of the manipulator. The method used to calculate the required actuator forces is as follows.

1. The feedback control law is implemented through the use of a set of spatial feedback vectors,  $\hat{F}_j$ , which are defined for each link in the system, and scalar feedback functions,  $\hat{\eta}_j$ , which are defined for each joint of the manipulator.
2. The  $\tau_j$  are computed using the  $\hat{F}_j$  and  $\hat{\eta}_j$  as in the following recursive equations:

$$\begin{aligned} \hat{f}_j &= \hat{F}_j + \hat{f}_k + \hat{f}_l \\ \hat{\rho}_j &= s'_j \hat{f}_j + \hat{\eta}_j \\ \tau &= E(Q)^T \hat{\rho} \end{aligned}$$

where  $k$  and  $l$  are the immediate descendants of link  $j$ ,  $\hat{\rho}$  is an  $(m + r) \times 1$  vector of the  $\hat{\rho}_j$ .

These types of calculations are common to a variety of controller algorithms and are included only for the users convenience. The choice of the  $\hat{F}_j$  and  $\hat{\eta}_j$  dictates the character of the controller being implemented. The following example is for an inverse dynamics controller.

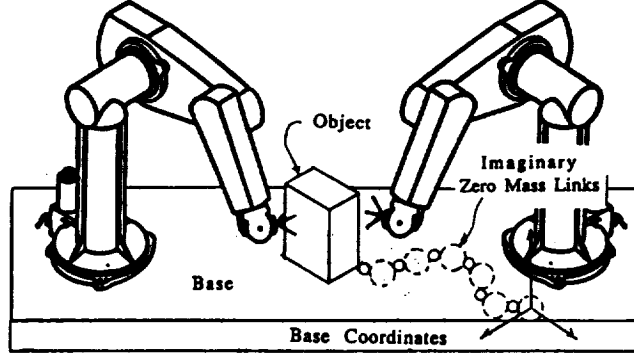


Figure 4: Example System Used in Simulation

In the inverse dynamics controller, both the spatial feedback vectors and feedback functions are used. The following variables are used to compute them.

$$\begin{aligned} Q_e &= Q_d - Q & \ddot{q} &= E(Q)\ddot{Q} + \dot{E}(Q)\dot{Q} \\ \dot{Q} &= \dot{Q}_d + K_d\dot{Q}_e + K_p\dot{Q}_e & \hat{v}_0 &= 0 \\ \dot{q} &= E(Q)\dot{Q} & \hat{v}_0 &= 0 \end{aligned}$$

where  $K_d$  and  $K_p$  are positive constants.

For  $i > 0$ , the spatial acceleration vectors are computed using recursive equations:

$$\begin{aligned} v_i &= \begin{cases} v_{\phi_i} + s_i \dot{q}_i & \text{if link } i \text{ is a successor} \\ v_{\phi_i} & \text{if link } i \text{ is a connector} \end{cases} \\ \dot{v}_i &= \begin{cases} \dot{v}_{\phi_i} + v_i \times s_i \dot{q}_i + s_i \ddot{q}_i & \text{if link } i \text{ is a successor} \\ \dot{v}_{\phi_i} & \text{if link } i \text{ is a connector} \end{cases} \end{aligned}$$

The spatial feedback vectors are functions of these vectors. For  $i > 0$ :

$$\hat{F}_i = \begin{cases} \hat{I}_i \dot{v}_i + v_i \times \hat{I}_i v_i & \text{if link } i \text{ is a successor} \\ 0 & \text{if link } i \text{ is a connector or terminator} \end{cases}$$

The scalar feedback functions are:

$$\hat{\eta}_i = \begin{cases} \hat{d}_i \dot{q}_i & \text{if link } i \text{ is a successor} \\ 0 & \text{if link } i \text{ is a connector} \end{cases}$$

The hat over the  $\hat{d}_i$  and the  $\hat{I}_i$  indicates they are only estimates of the corresponding components of the actual system. Thus,  $\hat{I}_i$  is the spatial moment of inertia matrix of link  $i$  using the estimated values of the link inertial parameters.

This control law is a recursive implementation of the following function:

$$\tau = H(Q)\ddot{Q} + C(Q, \dot{Q})\dot{Q}$$

## 4 Simulation Results

A simulation was performed of a two armed robot consisting of two PUMA 560 manipulators mounted on a single base. The objective of the simulation is to test the ability of the manipulators to grasp a floating object using the above inverse dynamics control algorithm. Each manipulator is equipped with an end-effector having three rigid fingers. There are three distinct phases of this task: approach, contact, and coordination. The approach phase is when both manipulators are approaching the object, but neither has contacted the object. The contact phase is the time after the first contact is made by either manipulator and before the coordination phase. The coordination phase is after both manipulators have made three point contacts with the object. A relatively low position feedback gain at each joint of the manipulators to provide the positional compliance required during contact with the object.

The system is illustrated in figure 4. To model the floating object, five massless links were inserted between the base coordinate frame and the object coordinates. These are denoted by the five large dotted line circles in the figure.

As expected the controller performed fairly well during the approach phase. Because of the small gains the settling time was about one second. This could be improved by increasing the gains with the result of reducing the compliance during the contact and coordination phases. The controller did not perform as well during the contact and coordination phase. Significant position errors and forces were observed due to the interaction of the object with the manipulator.

## 5 Conclusion

This paper presented an efficient method of simulating the motion of space based robots. The simulation includes not only the dynamics of the manipulators but also the dynamics of the objects in which the manipulators come into contact. Thus, the effect the robots have on the world in which they are a part is an important aspect of the simulation.

An example simulation was provided of a two armed robot grappling a satellite. It was found that the inverse dynamics controller worked reasonably well during the approach phase of this task, but it did not work very good during the contact and coordination phases. The controller was not successful in obtaining a stable grappling of the object. The conclusion is that a different control law is required for the three distinct phases of this task: approach, contact, and coordination. A position control should be used during the approach phase. A compliant motion control should be used during the contact phase and a dual-arm coordinated motion control should be used to manipulate the satellite.

In our simulation and the real robot controller we have the capability of switching between any one of a number a control laws from one sample period to the next. The problem we face is in deciding when a new control law should used. This is a complex decision in which several factors must be considered. The decisions for the switching times and the selection of the control laws should be based upon the task definition and available sensory measurements, such as tactile and force sensors. Except in a very simple manner, we currently have no way of either simulating this decision process or implementing it in the real robot controller. We believe the complexity dictates the use of artificial intelligence techniques such as expert systems to make these decisions. The lack of this decision making component is the greatest limitation of our system.

Finally, the last phase of controller development is the installation and testing with the real robot. This process is facilitated in the our laboratory by programming both the simulation and the real robot controller in the same language, Ada. Also, the definition of the spatial and scalar feedback functions are incorporated into the simulation as well as the real controller with identical pieces of Ada software. Thus, once the simulation results are acceptable, the controller code from the simulation is transfered to the robot's control computer, where it is recompiled and linked into the robot controller software. Future goals will be to integrate an expert system into both the simulation and the real robot motion controller to handle the control law switching decisions.

## References

- [1] J. Denavit and R. Hartenberg. A kinematic notation for lower-pair mechanisms based on matrices. *ASME Journal of Applied Mechanics*, pages 215-221, June 1955.

- [2] R. Featherstone. The calculation of robot dynamics using articulated-body inertias. *The Int. J. of Robotics Res.*, 2(1):13-30, Spring 1983.
- [3] R. Featherstone. *Robot Dynamics Algorithms*. Kluwer Academic Publishers, Norwel, MA, 1987.
- [4] E. G. Gilbert, D. W. Johnson, and S. S. Keerthi. A fast procedure for computing the distance between complex objects in three-dimensional space. *IEEE Journal of Robotics and Automation*, 4(2):193-203, April 1988.
- [5] R. Lathrop. Constrained (closed-loop) robot simulation by local constraint propagation. In *I.E.E.E. Int'l Conference on Robotics and Automation*, San Francisco, CA, 1986.
- [6] J. Y. S. Luh and Y. F. Zheng. Computation of input generalized forces for robots with closed kinematic chain mechanisms. *I.E.E. Trans. on Systems, Man and Cybernetics*, RA-1(2):95-103, June 1985.
- [7] Y. Nakamura and M. Ghodoussi. A computational scheme of closed link robot dynamics derived by d'alembert principle. In *IEEE Int'l Conf. on Robotics and Automation*, Raleigh, North Carolina, April 1988.
- [8] M. W. Walker. An efficient algorithm for the adaptive control of a manipulator. In *IEEE Int'l Conf. on Robotics and Automation*, Raleigh, North Carolina, April 1988.
- [9] M. W. Walker. Manipulator kinematics and the epsilon algebra. *I.E.E.E. Journal of Robotics and Automation*, 4(2):186-192, April 1988.
- [10] M. W. Walker and D. E. Orin. Efficient dynamic computer simulation of robotic mechanisms. *AME Journal of Dynamic Systems, Measurement and Control*, 104:205-211, Sept. 1982.

## PRELIMINARY RESULTS ON NONCOLLOCATED TORQUE CONTROL OF SPACE ROBOT ACTUATORS

Scott W. Tilley, Colin M. Francis, Ken Emerick, and Michael G. Hollars

Ford Aerospace, Space Systems Division  
Palo Alto, California 94303

### Abstract

In the Space Station era, more operations will be performed robotically in space in the areas of servicing, assembly, and experiment tending among others. These robots may have various sets of requirements for accuracy, speed, and force generation, but there will be design constraints such as size, mass, and power dissipation limits. For actuation, a leading motor candidate is a DC brushless type, and there are numerous potential drive trains each with its own advantages and disadvantages. This experiment uses a harmonic drive and addresses some inherent limitations, namely its backdriveability and low frequency structural resonances. These effects are controlled and diminished by instrumenting the actuator system with a torque transducer on the output shaft. This noncollocated loop is closed to ensure that the commanded torque is accurately delivered to the manipulator link.

The actuator system is modelled and its essential parameters identified. The nonlinear model for simulations will include inertias, gearing, stiction, flexibility, and the effects of output load variations. A linear model is extracted and used for designing the noncollocated torque and position feedback loops. These loops are simulated with the structural frequency encountered in the testbed system. Simulation results are given for various commands in position. The use of torque feedback is demonstrated to yield superior performance in settling time and positioning accuracy.

An experimental setup being finished consists of a bench mounted motor and harmonic drive actuator system. A torque transducer and two position encoders, each with sufficient resolution and bandwidth, will provide sensory information. Parameters of the physical system are being identified and matched to analytical predictions. Initial feedback control laws will be incorporated in the bench test equipment and various experiments run to validate the designs. The status of these experiments is given.

### 1. Introduction

There are a wide variety of applications in space that could be assisted or performed telerobotically. These missions include large space structure assembly, module changeouts, maintenance, inspection, and refueling. This paper will assume a simple generic mission has been chosen to generate reasonable, preliminary manipulator requirements. A preliminary, symmetric arm configuration consists of two links with 7 degrees of freedom [1]. Obviously, arm mass and power requirements are to be minimized. Manipulator requirements are then reflected in the actuator subsystem sizing and component selection. This research focuses on the details of one

single degree of freedom joint at one end of the arm. Manipulator and derived joint requirements are given in Table 1.

Direct drive actuators initially appear attractive for space robotics because the manipulator is not required to support itself or a payload. However, there are needs of sustained tip forces to accelerate (or decelerate) payloads and to apply insertion forces during module changeouts. A 20 lb insertion force at the reach of 80 inches implies a 1600 in-lb (180 N-m) torque at the shoulder joint plus some margin. The size and mass of a direct drive joint would be large and yield a robot system design that was prohibitively expensive to launch and probably not capable of withstanding the thermal environment of space due to the high power dissipation.

Geared drives have the advantage of being lighter, requiring less power, and being more compact than an equivalent direct drive. However, gearing introduces a new set of problems to be overcome including, but not limited to: lower efficiency, various types of friction, torsional flexibility, backlash, reliability and life considerations. These issues can be adequately resolved and most space robot applications will employ some type of gearing.

Manipulator		Joint	
Manipulator Reach	2 m (79 in)	Gear Ratio	200
Maximum Tip Speed	0.5 m/sec	Maximum Joint Rate	0.25 rad/sec
Tip Position Resolution	0.001 m (0.04 in)	Joint Position Resolution	0.5 mrad
Sustained Tip Force	90 N (20 lbf)	Sustained Joint Torque	180 N-m
Tip Force Resolution	0.9N (0.2 lbf)	Joint Torque Resolution	1.8 N-m

Table 1: Manipulator and Joint Requirements

The gearing type chosen should ameliorate the worst effects for the given mission requirements at the expense of other effects to be compensated for. For instance, spur gears are efficient, but introduce backlash. The reduction of backlash, however, introduces compliance and so on. Applicable gearing systems such as spur gears, planetary gears, harmonic drives and others have been studied [2,3]. Of these, harmonic drives possess the best combination of performance characteristics for a space robot. They provide high gear ratios in one pass, have zero backlash, and have acceptable stiffness, friction, and efficiency. They are in current use in terrestrial robots and have been successfully used in spaceflight actuators.

Harmonic drives do present some problems that must be addressed before their use in a dexterous space manipulator. Motor friction is multiplied through the gearing producing undesirable tip force breakaway levels and a lack of adequate backdriveability. Imperfections in the gearing also produce output position errors at a frequency of twice the motor speed. This can cause vibration as the motor speeds up and down in a maneuver and excites system resonances. The

harmonic drive dominates the manipulator compliance more than the links and this results in low system cantilever frequencies during large payload manipulations. The intelligent use of noncolocated torque feedback can drastically reduce these effects [4,5] by insuring that the joint actuator delivers commanded torque to the manipulator link. The servo control system must be designed to make the joint a linear device for applying torque. These loops will be first designed and simulated on a nonlinear joint model before being attempted in the digital control of the prototype joint.

## 2. Physical Actuator System

The testbed built includes the components required in a robot joint, but it is physically arranged to permit easy modification rather than represent an actual flight joint. The key elements of the testbed were chosen to meet the requirements set out previously in Table 1. The components and their nominal characteristics are presented in Table 2.

Motor	Type	DC, brushless
	Peak torque rating:	575 oz-in (4.06 N-m)
	Electrical time constant ( $\tau_E$ )	4 msec
	Motor torque constant	60 oz-in./amp
	No load speed	1800 RPM
	Rotor inertia ( $J_{m1}$ )	$5.8 \times 10^{-4}$ kg-m <sup>2</sup>
	Static friction	12 oz-in max
Input Bearings	Friction ( $B_{v1}$ )	2 oz-in max
Harmonic Drive	Gear ratio (N)	200:1
	Maximum torque output	2890 in-lb (327 N-m)
	Torsional stiffness (K)	100,000 in-lb/rad initially, then stiffens
	Wave generator inertia ( $J_{m2}$ )	$1.8 \times 10^{-4}$ kg-m <sup>2</sup>
	Starting torque	11 oz-in
Torque Transducer	Rated capacity	5000 in-lb (565 N-m)
	Resolution	1:5000
	Torsional stiffness	750,000 in-lb/rad
Output Bearings	Friction ( $B_{v2}$ )	40 oz-in max
Position Encoders	Resolution	1024 pulses per rev plus quadrature
	Frequency response	100 kHz

Table 2: Nominal Component Characteristics

The maximum speed and torque that the motor and harmonic drive will operate at during testing (still meeting slew requirements) is approximately one third of their rated capacity. The

various testbed transducers are adequate for meeting requirements. It is understood that a real system will have additional error sources such as misalignments, thermal distortions and others, but they are not addressed here. Highly precise end effector position and force measurements will ultimately require end point sensors and noncolocated end point control, whose benefits are being currently studied [6]. This does not detract from the significance of the noncolocated torque feedback loop. A brake is not currently used on the testbed joint because regenerative (dynamic) braking will be investigated in a parallel experiment.

### 3. Actuator System Model

A nonlinear model of the actuator plant containing the dominant physical phenomena is shown in Figure 1. The figure is a simple representation of the system and is not intended to reflect the physical layout of the joint. With the motor operated well below its no load speed, it will be capable of providing continuous demanded torque in the speed range used. The switching power amplifier used will not saturate under test conditions and it includes current feedback thus reducing back EMF effects.

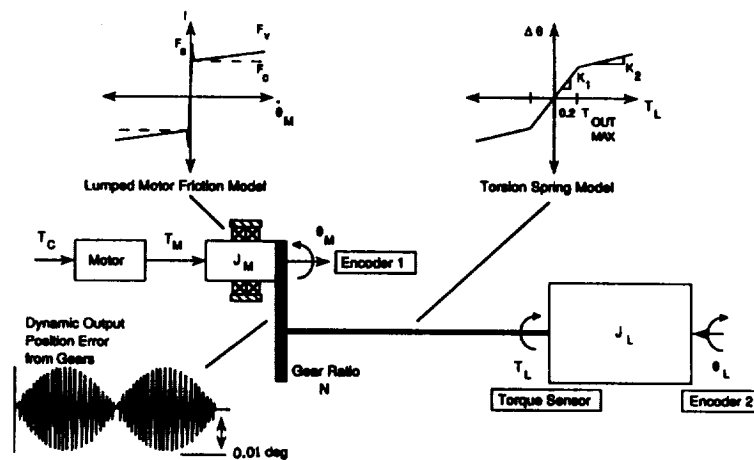


Figure 1: Nonlinear Model of Actuator

The dominant motor effect is inductance in the windings creating phase loss. The motor and wave generator inertias have been lumped together. Also, the friction model inboard of the harmonic drive gearing is lumped together into a static, Coulomb, and viscous friction model which will degrade output torque response when multiplied through the gearing. This friction model will be very difficult to verify experimentally and may change with component aging, therefore it is essential that the torque feedback loop be robust enough to handle a range of frictions. The position error at twice the motor speed due to gearing imperfections is modelled as a forcing disturbance torque. The compliance in the harmonic drive is modelled as a piecewise linear, stiffening spring. The mechanism stiffens as torque is applied because more surface area of the gear teeth are forced into contact. The cup, torque transducer, and load inertia are lumped together. Notice the output shaft rotation is opposite to the input shaft rotation. The sensor signals available are the motor



position  $\theta_M$  (digital), output shaft position  $\theta_L$  (digital) and torque  $T_L$  (analog). Additional phase loss will be introduced into the system through antialiasing filters and any differencing of position signals to get rate without tachometers. These key parameters are identified in vendor literature, but must be measured on the physical hardware in the testbed.

A simplified, linear model can be extracted by neglecting nonlinear friction and using one stiffness value and this will yield equations of motion (1), (2), and (3). Torque sensed at the load is due to spring and external ( $T_E$ ) torques. From these, the colocated and two noncolocated transfer functions are derived and given in (4), (5), and (6). The numerical values in the transfer functions are based on the nominal testbed load inertia,  $J_L$ , of  $3.0 \text{ kg-m}^2$  and a small amount of viscous bearing friction at the motor and output shaft (2 and 10 oz-in per rad/sec, respectively). The pole-zero patterns in the noncolocated transfer functions can easily destabilize a feedback system. Also note that output torque on the load cannot be maintained (zero DC gain) without an external torque. The system will simply spin up to a steady state speed (no load speed) where torque can no longer be generated at the output.

$$\tau_E \dot{T}_M + T_M = T_C \quad (1)$$

$$J_M \ddot{\theta}_M + B_{V1} \dot{\theta}_M + \frac{K}{N} (\frac{\theta_M}{N} - \theta_L) = T_M \quad (2)$$

$$J_L \ddot{\theta}_L + B_{V2} \dot{\theta}_L + K (-\frac{\theta_M}{N} + \theta_L) = T_E \quad (3)$$

and

$$\frac{\theta_M(s)}{T_C(s)} = \frac{(3.29 \times 10^5)(s^2 + 61.1^2)}{s(s + 16.9)(s + 250)(s^2 + 1.53s + 63.8^2)} \frac{\text{rad}}{\text{N-m}} \quad (4)$$

$$\frac{\theta_L(s)}{T_C(s)} = \frac{(6.14 \times 10^6)}{s(s + 16.9)(s + 250)(s^2 + 1.53s + 63.8^2)} \frac{\text{rad}}{\text{N-m}} \quad (5)$$

$$\frac{T_L(s)}{T_C(s)} = \frac{(1.84 \times 10^7)s}{(s + 16.9)(s + 250)(s^2 + 1.53s + 63.8^2)} \frac{\text{N-m}}{\text{N-m}} \quad (6)$$

This analysis can be modified and applied to the case where the load inertia is constrained from moving and torque is simply transmitted to the environment. This simulates a manipulator in contact with a fixed object and assumes a rigid link (arm). By setting the load angle and its derivatives to zero (or making the load inertia extremely large) in equations (1) through (3), the joint equations of motion for applying force to a fixed surface become evident. The resulting transfer functions are equations (7) and (8).

$$\frac{\theta_M(s)}{T_C(s)} = \frac{(3.29 \times 10^5)}{(s + 250)(s^2 + 18.4s + 19.2^2)} \frac{\text{rad}}{\text{N-m}} \quad (7)$$

$$\frac{T_L(s)}{T_C(s)} = 56 \frac{\theta_M(s)}{T_C(s)} \frac{\text{N-m}}{\text{N-m}} \quad (8)$$

A physical manipulator will attach to and move payloads. This creates large changes in the apparent inertia of the arm. It is instructive to look at the magnitude of the cantilever and free-free resonances and their relative separation as loads vary. The load inertia is matched to the motor and gearing through the square of the gear ratio. Table 3 below shows a range of frequencies with various load inertias. Asymptotically as the outboard load increases, the free-free frequency will approach the cantilever frequency in the matched case. This is a case of the tail wagging the dog as the motor cantilevers while the load remains stationary. Colocated proportional-derivative controller historically used in servo controls usually perform no better in bandwidth than about half the cantilever frequency [6].

Outboard Inertia (kg-m <sup>2</sup> ) and [matched inertia ratio $J_L/J_M N^2$ ]	Frequencies (rad/sec)	Comment
3 [0.1]	$\omega_c = 61.1, \zeta=0.00$ $\omega_f = 63.8, \zeta=0.01$	Testbed range
30 [1.0]	$\omega_c = 19.3, \zeta=0.00$ $\omega_f = 25.7, \zeta=0.16$	Matched case $\omega_f = \omega_c \sqrt{2}$
60 [2.0]	$\omega_c = 13.7, \zeta=0.00$ $\omega_f = 21.8, \zeta=0.26$	Unloaded manipulator arm
600 [20.0]	$\omega_c = 4.30, \zeta=0.00$ $\omega_f = 19.3, \zeta=0.45$	Manipulator with payload

Table 3: Structural Frequency Variations with Load Inertia Variations

#### 4. Control Design and Simulation

Control analysis is performed to yield a suitable feedback controller to meet the requirements of Table 1, especially in position and force resolution at the tip. Simple root locus and Linear Quadratic Gaussian (LQG) techniques are used to derive compensator transfer functions for both position and torque feedback [7]. Output feedback will be used, not full state feedback [8]. The closed loop results for small slews in position are then evaluated for further refinement of the control algorithm. All controllers designed will operate within the actuator torque and bandwidth capabilities.

First, a colocated position feedback loop is derived. The open loop transfer function, equation (4), is dominated by the rigid body poles. A simple lead filter is chosen. The resonant mode is effectively

trapped by the cantilever zero. Physically, the mode is difficult to observe through the motor angle and difficult to actively damp. An angle slew at 0.25 rad/sec representing 20 cm of tip motion is performed using the full nonlinear simulation, see Figure 2. Load angle (synonymous for joint angle) is commanded by simply commanding a motor angle multiplied by the gearing. Notice the undesirable ringing in the load after the motor shaft has locked up under stiction. The load motion no longer has access to the energy dissipation mechanisms in the motor and relies on outboard structural and bearing damping alone. Joint position settles slowly and meeting requirements may not be possible without accurate knowledge of the stiction levels. Torque commands are also difficult to achieve across the joint due to stiction and the nonlinear spring. Notably, a minimum tip force of 17.5 N (3.9 lbf) is needed to break motor stiction.

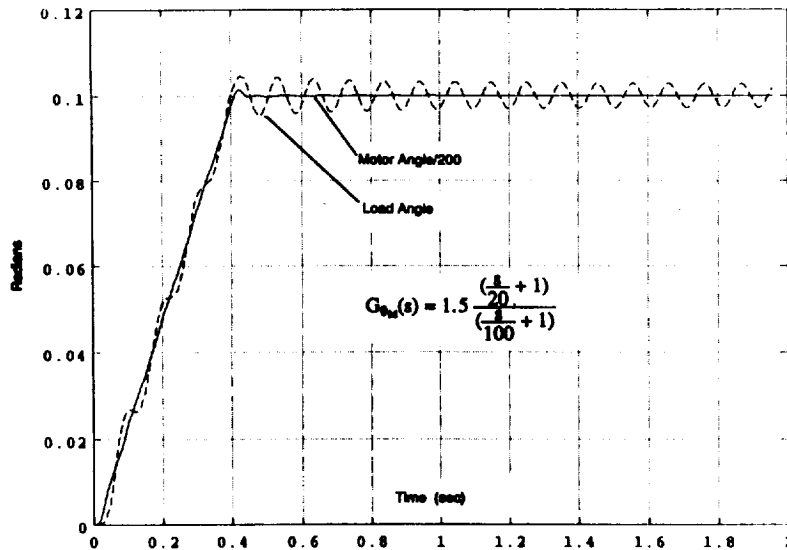


Figure 2: Position Slew With Colocated Angle Feedback

Second, a noncolocated position feedback loop is derived. The open loop transfer function, equation (5), is first approximated by the rigid body poles and the resonance while ignoring the motor inductance. A reduced order compensator is designed using LQG regulator techniques with output weighting. A root locus of this compensation with the linear plant model is given in Figure 3. In the absence of the cantilever zeroes, the resonances can be actively damped at the expense of increased motor activity. This compensation was applied to the nonlinear simulation.

This closed loop system is slewed using the full nonlinear plant model simulation, see Figure 4. The load angle no longer rings although the rigid body performance is slightly slower. The steady state error due to motor stiction for this compensator is still above the position requirement. Increased compensator gain is necessary, but limit cycling quickly occurred with higher gains. Also, there exists the potential for control spillover with higher gain. Finally, this loop cannot be used for controlling output torque levels when the joint is in contact with the environment as the load angle is fixed.

Thus far, the position controllers have failed to provide adequate servo performance regardless of the feedback sensor location. Perfect knowledge of the plant dynamics and parameters can yield

feedforward compensation, but feedback techniques are preferred for robustness. Output torque feedback is the solution. This may be achieved through successive loop closure techniques or full multi-input multi-output (MIMO) LQG design. Both are tried here.

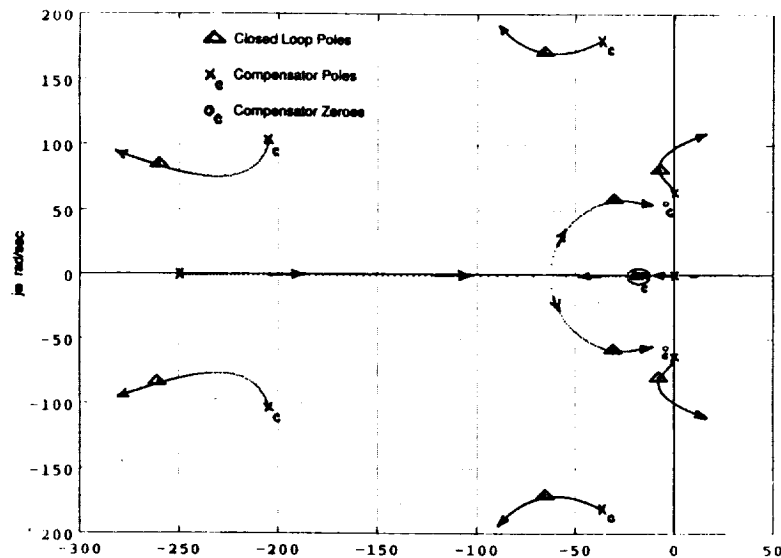


Figure 3: Noncolocated Position Feedback Root Locus

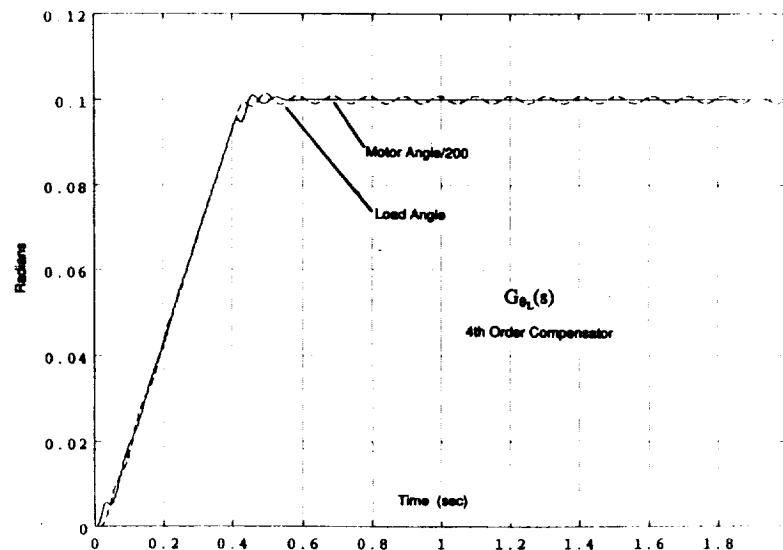


Figure 4: Position Slew With Noncolocated Position Controller

First, a simple lead filter stabilizes the noncolocated torque loop and both actively damps the load resonance and reduces the friction effects observed on the load side of the gearing. Next, the simple colocated position loop is closed around the torque inner loop. This produced good position performance during the slew shown in Figure 5. This design also yielded good torque response with the joint in contact for torques that did not exceed the first linear region of the spring. Higher torque commands resulted in instabilities due to the higher effective loop gain. Finally, both noncolocated

ORIGINAL PAGE IS  
OF POOR QUALITY

loops (position and torque) are closed simultaneously using LQG techniques. This controller regulates load angle well and effectively damps vibration during the slew in Figure 6.

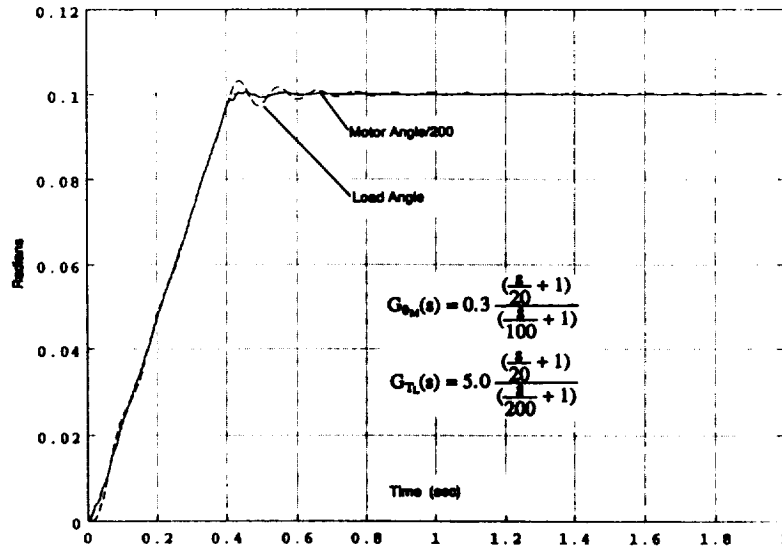


Figure 5: Position Slew With Colocated Position Control With Inner Torque Loop

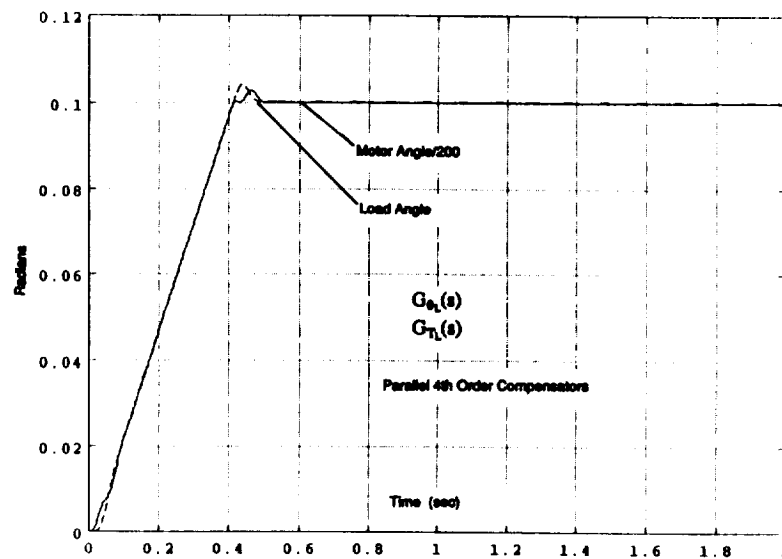


Figure 6: Position Slew With MIMO Controller

## 5. Summary

To summarize these results, a comparison is made of the position controllers with and without torque feedback and is shown in Figure 7. Torque feedback usage yields superior results over either colocated or noncolocated feedback used alone. Slew tracking errors are diminished and damping is improved thus reducing settling times. The qualitative results from this research are valid, although quantitative measures are difficult to extract as the controllers are not "normalized" to each other in terms of DC gain.

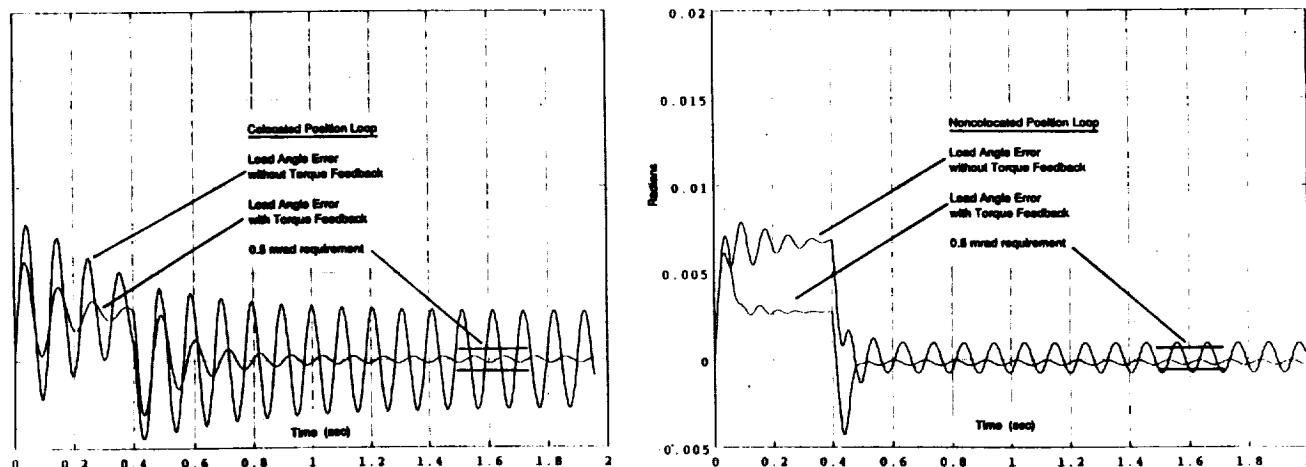


Figure 7: Servo Performance With and Without Inner Torque Loop

The nonlinear actuator model and control algorithms will be validated through hardware test as soon as the joint testbed is complete. Further analysis is also needed to create designs that provide robust torque control when the joint in contact (load is stationary). Parameter sensitivity studies and nonlinear limit cycle analysis using describing functions are planned to further investigate the spring stiffening and friction effects.

## 6. References

- [1] Gretz, B. and Tilley, S., "Kinematics, Controls, and Path Planning Simulation Results for a Redundant Manipulator". In proceedings of NASA Conference on Space Telerobotics, Pasadena, California, January , 1989.
- [2] Chun, W. and Brunson, P., "Actuators For A Space Manipulator". In proceedings of the Conference on Space Applications of AI and Robotics, NASA Goddard, May 13-14, 1987.
- [3] Holzbock, W.G., Robotic Technology Principles and Practice. Published by Van Nostrand Reinhold Company, New York, New York, 1986.
- [4] Wu, C. H. and Paul, R. P., "Manipulator Compliance Based on Joint Torque Control". In proceedings of the 19th IEEE Conference on Decision and Control, Vol. 1, pp. 84-88, Albuquerque, New Mexico, December, 1980.
- [5] Pfeffer, L., Khatib, O., and Hake, J., "Joint Torque Sensory Feedback In The Control Of A PUMA Manipulator". In proceedings of the American Control Conference, pp. 818-824, Seattle, Washington, June, 1986.
- [6] Hollars, M. G., Experiments in End-Point Control of Manipulators with Elastic Drives. PhD Thesis, Department of Aeronautics and Astronautics, Stanford University, May, 1988.
- [7] Franklin, G.F., Powell, J.D. and A. Emami-Naeini, Feedback Control of Dynamic Systems. Published by Addison-Wesley Company, Reading, Massachusetts, 1986.
- [8] Won, S.C., Lim, D.J. and D.H.Chyung, "DC Motor Driven Robotic Manipulator Control". In proceedings of 24th IEEE Conference on Decision and Control, Ft. Lauderdale, Florida, December, 1985.

## **PORTABLE DEXTROUS FORCE FEEDBACK MASTER FOR ROBOT TELEMANNIPULATION (P.D.M.F.F.)**

Grigore C. Burdea  
Electrical and Computer Engineering Dept.  
Rutgers University  
Piscataway, NJ 08855-0909

Thomas H. Speeter  
A.T.&T. Bell Laboratories  
Machine Perception Research Dept.  
Holmdel, NJ 07733

### **Abstract**

A major drawback of open loop masters is a lack of force feedback, limiting their ability to perform complex tasks such as assembly and repair. We present a simple dextrous force feedback master for computer assisted telemanipulation. The device is compact, portable and can be held in the operator hand, without the need for a special joystick or console. The system is capable of both position feed forward and force feedback, using electronic position sensors and a pneumatic micro actuator. The level of forces exercised by the pneumatic actuator is such that near rigidity may be attained. We present experimental results showing good system linearity and small time lag.

### **1. Introduction**

Present telemanipulation techniques include mechanical masters, open loop servo masters, and to a lesser extent closed loop servo masters. Direct mechanical telemanipulation is often the simplest method, but cannot be used in applications where the slave is not in the immediate vicinity of the master, as may well be the case in space applications. Closed loop servo telemanipulation eliminates the proximity requirement, but when force feedback is provided by servo encoders on the slave arm, it is necessary to have two nearly identical devices to act as master and slave. This duplication of resources may be prohibitive in terms of cost and payload weight.

Current research efforts aim at eliminating the duplicate master by replacing it with force feedback joysticks or sensorized spheres[3]. These devices are less "natural" to use by an operator since direct similitude does not exist between human hand and robot finger motions.

Dextrous master control represents a recent addition to the field of telemanipulation. A dextrous master can replace the classical manipulator arm, joystick or keypad master with an operator's hand motions[6]. Use of the human hand is a natural form of control and is applicable to both nondextrous and dextrous slave devices. Because the human hand is used as master, duplication of hardware is not required, and weight, inertia and friction are reduced. This can bring significant improvements in the time necessary to com-

plete a task. It is estimated [2] that an improvement on the order of 10 can be expected on the time efficiency quotient[7] when a dextrous master is used in place of keypad control.

While an open-loop dextrous master creates a natural control environment, it lacks the ability to bring force feedback to the operator hand, which in turn limits the utility of the slave device. This paper describes work towards the development of a dextrous master with force feedback. Such a device will allow the execution of complex tasks such as assembly and repair, using the human hand as master in closed loop teleoperation environment.

## 2. Conceptual design

The design of the Portable Dextrous Master with Force Feedback (PDMFF) was guided by the need to bring force feedback from a robot end effector to the human hand serving as master. The aim is to produce a compact, hand held device that fits inside the palm. It should function as position controller for the robot end effector (either conventional gripper or dextrous hand), and provide force feedback to the operator.

We have developed an initial master device to control and exert feedback in one degree of freedom. The position sensing device is an LVDT transducer and force feedback is provided by a small pneumatic piston in parallel with the LVDT (Fig. 1). The LVDT requires only two contact points to secure its ends, therefore two fingers are sufficient to hold it. Use of the thumb and middle fingers assures a good grip and sufficient distance to accommodate the LVDT and piston.

The first joint of the thumb has two degrees of freedom namely anteposition/retro-position and abduction (Fig. 2) [1]. Human factor studies[5] show that the most comfortable thumb postures are those with little anteposition. The need to minimize fatigue implies that such postures should be accommodated by the master. A sphere joint on the middle finger mount allows for 70 degrees rotation. This combined with the translation movement of the active element produces a conic work envelope as shown in Fig. 3 and allows the user to comfortably position the PDMFF between his fingers.

The volume of the master work envelope is given by:

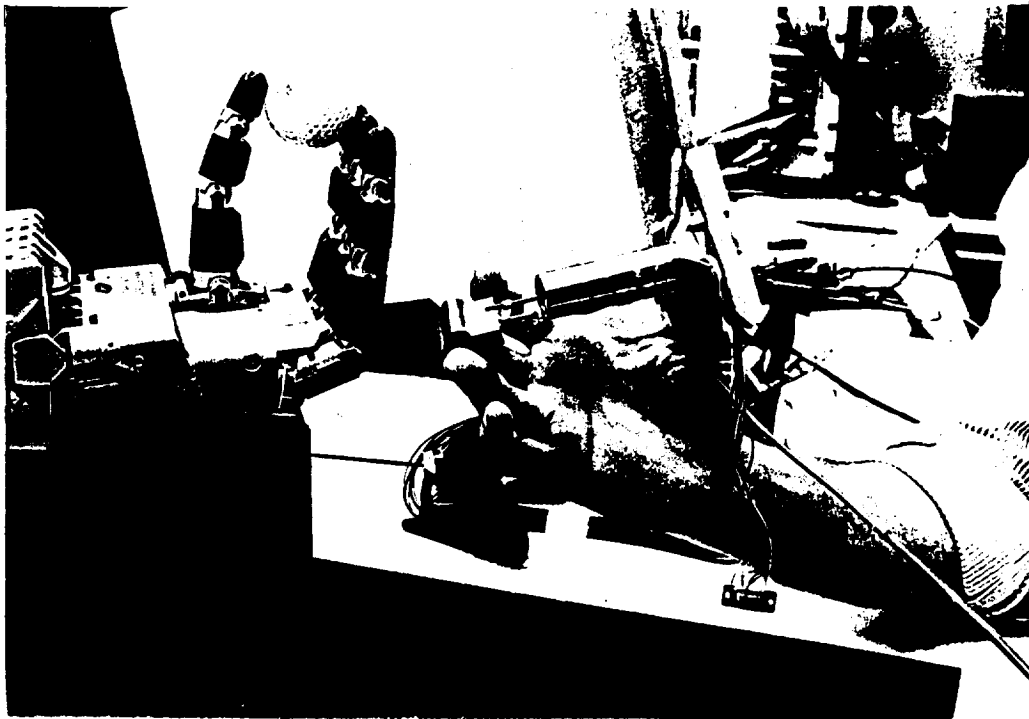
$$\pi \sin^2 \alpha \cos \alpha \left( \frac{d^2}{3} + ld + l^2 \right) \quad (1)$$

where:

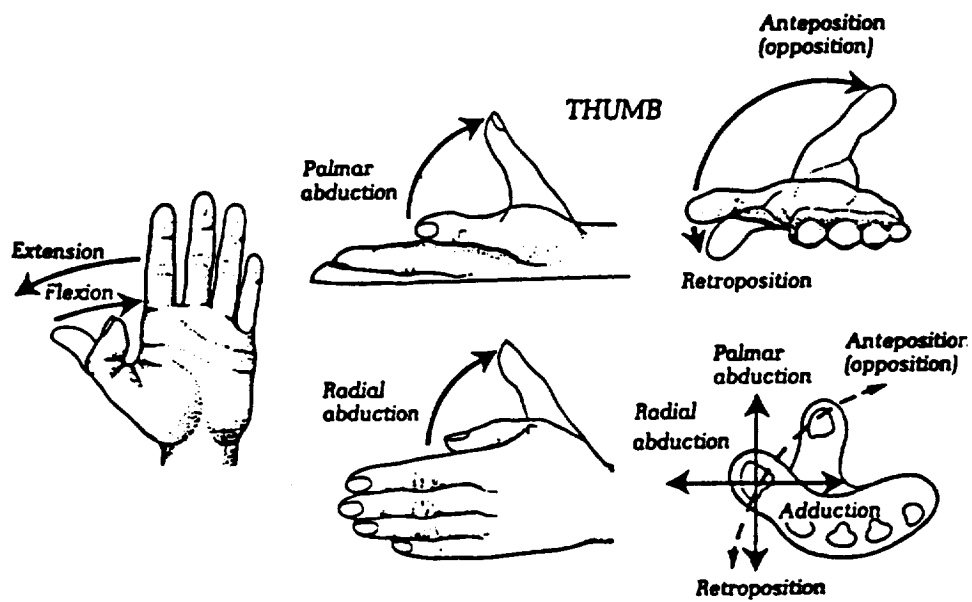
- $\alpha$  is the sphere joint angle,
- $d$  is the linear travel,
- $l$  is the length of the mechanical mount.

Due to limitations on the thumb's range of motion, the useful volume used to control the slave gripper is about 25% of the work envelope.

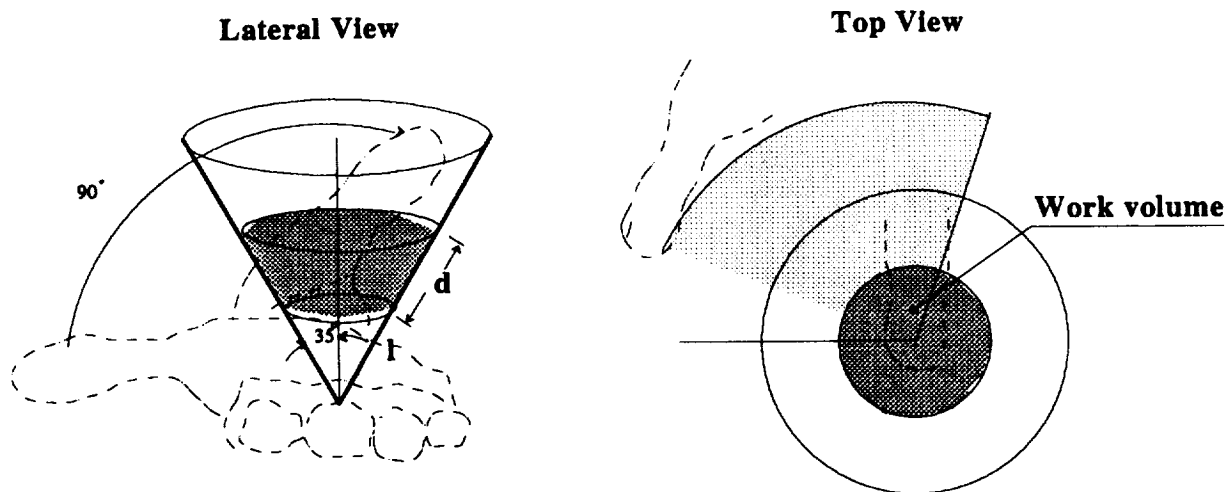




**Fig. 1 - PDMFF prototype**

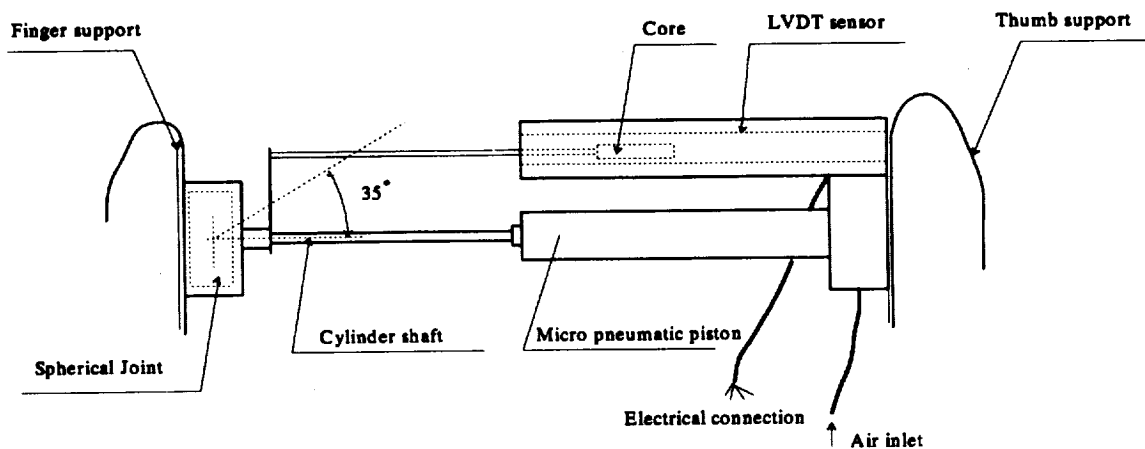


**Fig. 2 - Terminology of thumb motion [1]**



**Fig. 3 - PDMFF work envelope**

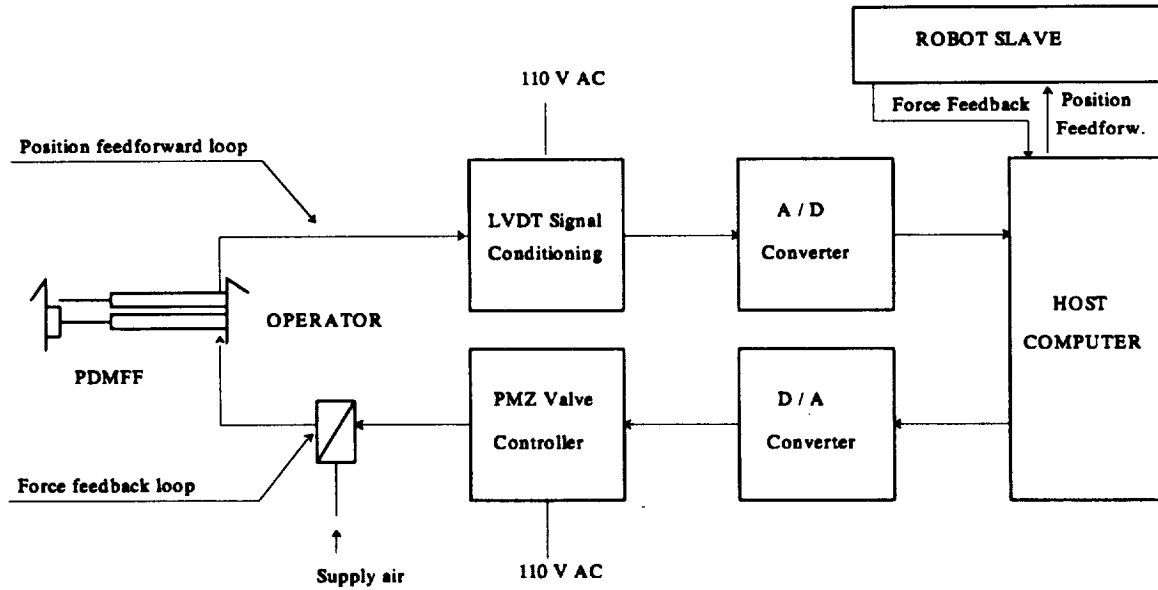
The desire for reduced dimensions and weight, while maintaining sufficient force capability led to the selection of a pneumatic micro cylinder as the active element. The cylinder spring is removed, allowing for free motion of the piston. The piston shaft is coupled with a second shaft attached to a magnetic core traveling inside the LVDT sensor. The resulting configuration is shown in Fig. 4.



**Fig. 4 - PDMFF design**

### 3. Experimental installation

The PDMFF is integrated with a host computer for control as shown in Fig. 5.



**Fig. 5 - Experimental Installation**

Position signals from the LVDT are passed through a signal conditioning unit, giving an analog signal (voltage) proportional to the core's displacement. This voltage is digitized by an analog to digital converter and sampled by a host computer at 500Hz. The host uses this signal to position the slave robot, in this case the Utah-MIT hand[4]. Force feedback from the hand is sampled by the host and used to drive the pneumatic cylinder. At this point a scaling up or down may be implemented to adjust the gain in the feedback loop. A voltage is sent to the valve controller which raises or lowers the air pressure in the cylinder using an analog proportional control system.

### 4. Test results

PDMFF was tested both under static and dynamic conditions. The static test was aimed at determining the force exercised by the cylinder as a function of the voltage controlled by the host computer. Four tests were run with air supply pressures,  $P_{air}$ , of 60, 70, 80 and 90 psi. For each supply pressure voltages from 0 to 10 V were applied to the valve controller. Under these conditions the force was measured with a load cell. The results are presented in Fig. 6. The force feedback is given by:

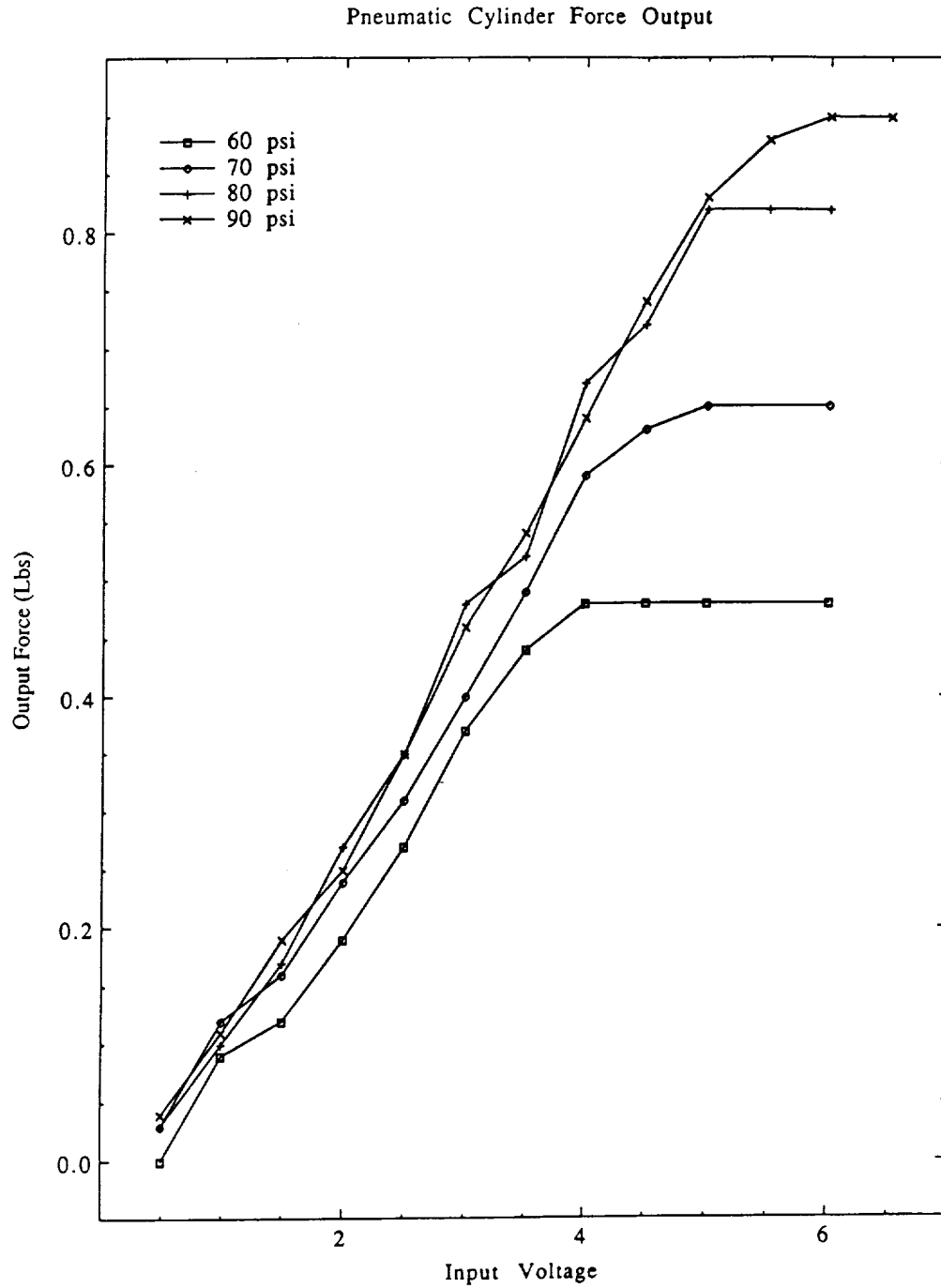
$$F_{feedback} = P_{air} A - F_{friction} \quad (2)$$

where:

$A$  is the piston area,

$P_{air}$  is the air pressure after the valve regulator,

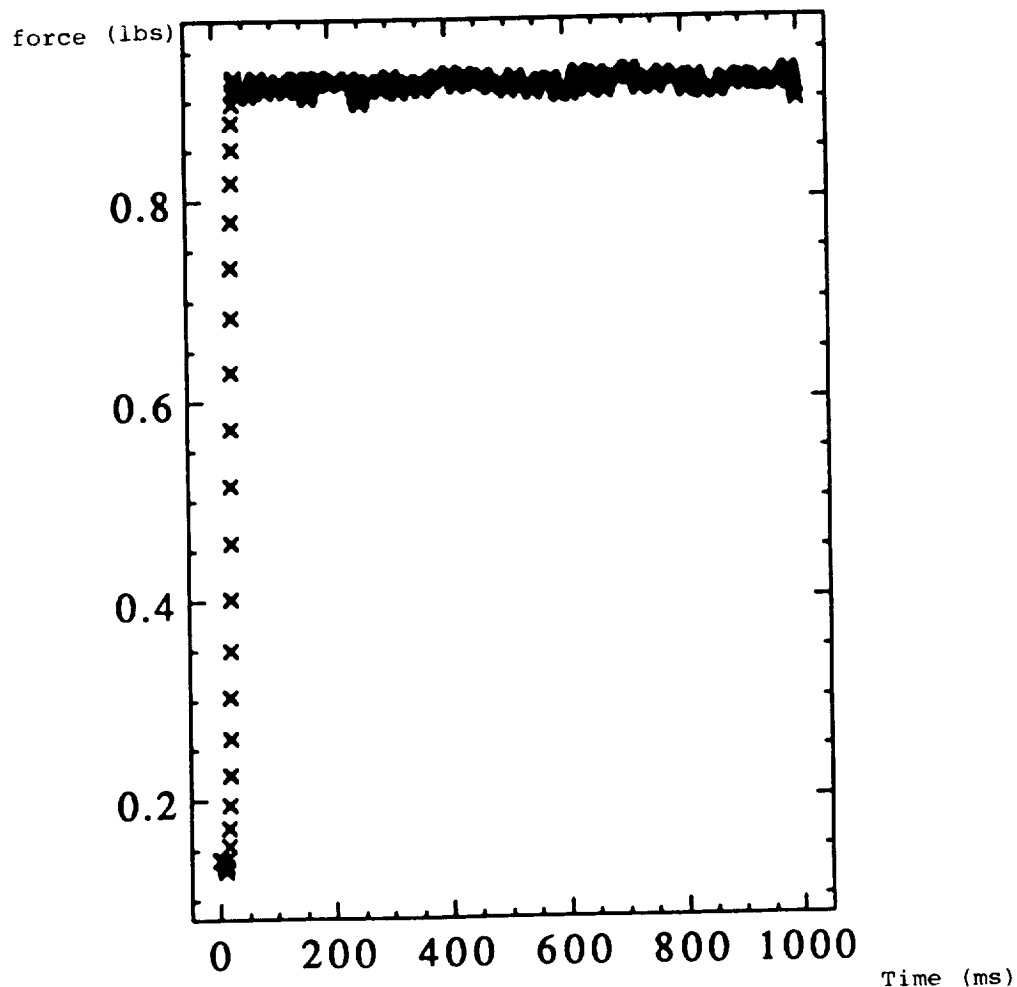
$F_{friction}$  is the piston friction.



**Fig. 6 - Micro cylinder force measurements**

The linearity of equation (2) is reflected in the graph, with the exception of plateaus appearing at higher voltages. At these plateaus maximum air flow is applied on the piston, and force no longer varies with increased voltage. A supply pressure of 90 psi and a voltage of 6V results in almost a pound force. This is a large enough force to produce near rigidity in the master and is sufficient to simulate the sensation of a solid object in the grasp.

A second set of tests measured the system response to a step function applied to the master. The same load cell was used to determine the force variation in time. Results are presented in Fig. 7. Tests showed an approximately 50 msec rise time with almost no overshoot or oscillation.



**Fig. 7 - Dynamic response for step function input**

## 5. Integration with robotic slaves

The design of the PDMFF makes it compatible with both dextrous and nondextrous end effectors. When coupled with a parallel finger gripper, the LVDT signal controls the gripper opening in a linear relationship. Since the piston travel is fixed for a given cylinder, the gripper opening is given by

$$L_{gripper} = LVDT_{signal} \frac{G}{d} \quad (3)$$

where

$G$  is the gripper maximum opening,  
 $d$  is the piston maximum extension.

When telemanipulating with a dextrous hand, we are faced with the problem of controlling multiple degrees of freedom with a device with only one degree of freedom. As an initial test we have used the PDMFF to control a three-fingered pinching motion of the Utah/MIT dextrous hand. An initial hand pose, with the fingers open and a final position with the thumb, index and middle fingers in a pinching posture are established. The position of the core within the LVDT is used to linearly interpolate between the initial and final pose and the hand is driven to the desired positions by the hand's resident position servo system. The resulting motion, as the PDMFF is opened and closed is a natural pinch and release action of the hand.

Forces exerted on objects by the hand are determined from the tensions sensed in the tendons of the hand. Each joint of the hand is operated by two tendons, one for flexion and one for extension, and each tendon is gauged, allowing their tension to be monitored. By observing the difference between flexion and extension tensions in the thumb, the net force exerted on an object can be deduced. This value is scaled and used to directly drive the valve of the pneumatic piston. The piston's internal pressure, therefore, is directly proportional to the force being exerted by the hand on the grasped object.

Objects of various shapes and sizes can easily be grasped and held by the dextrous hand using the PDMFF as master. Forces reflected from the hand to the user allow deft control of the grasping force. After a short learning period, the user is able to grasp objects lightly, at the verge of dropping, or with great force, greater than capable with the human hand alone by taking advantage of the strength of the Utah/MIT dextrous hand. The rigidity or compliance of an object grasped by the dextrous hand can be felt by the user in a very natural way, by squeezing the object and feeling the resulting reflected forces.

## 6. Conclusion

The force reflecting master described here allows the user to control the motion of, and feel the force exerted on a slave robotic device. The PDMFF uses a simple parallel arrangement of an LVDT position sensor and micropneumatic piston to provide a closed

loop telemanipulation system.

Interfaces to the PDMFF are simple, consisting of one voltage out, corresponding to the LVDT's position, and one voltage in, driving the pneumatic cylinder. Control of the slave will vary in complexity depending upon the device, however the simplicity of the PDMFF may allow the use of discrete components in the interface between master and slave rather than requiring the intervention of a processor. This can further reduce the complexity, cost and size of a master-slave system using the PDMFF, and may allow the development of compact, multidegree of freedom masters using several PDMFF-like mechanisms in parallel.

## 7. Acknowledgements

Work on this paper has been supported by funds from Rutgers University Research Council Grant and from AT&T Bell Laboratories.

## References

1. American Society for Surgery of the Hand, *The Hand: Examination and Diagnosis*, Churchill Livingstone, London (1983).
2. Burdea, Grigore C. and Kicha Ganapathy, "On Dextrous Robot Telemanipulation and Learning," AT&T Bell Laboratories Technical Report 11352-881003-11TM.
3. Hirzinger, G., "The Space and Telerobotic Concepts of DFVLR Rotex," *1987 Proceedings IEEE International Conference on Robotics and Automation*, pp. 443-449, IEEE, (1987).
4. Jacobsen, S.C., J.E. Wood, D.F. Knutti, and K.B. Biggers, "The Utah-MIT Dextrous Hand: Work in Progress," *Journal of Robotics Research* 3 (4), pp. 21-50 (1984).
5. McCormick, E.J., *Human Factors in Engineering and Design*, McGraw-Hill (1976).
6. Pao, Lucy Y. and Thomas H. Speeter, "Transformation of Human Hand Positions for Robotic Hand Control," *1989 IEEE International Conference on Robotics and Automation*, IEEE, (1989). (submitted)
7. Vertut, Jean and Philippe Coiffet, "Teleoperation and Robotics: Applications and Technology," *Robot Technology* 3B, 256 pp., Prentice Hall, (1986).





# Experiences with the JPL Telerobot Testbed

## — Issues and Insights —

Henry W. Stone, Bob Balaram, and John Beahan

Tele-Autonomous Systems Group  
Jet Propulsion Laboratory  
California Institute of Technology  
Pasadena, CA 91109

### Abstract

JPL's Telerobot Testbed [7] is an integrated robotic testbed used to develop, implement, and evaluate the performance of advanced concepts in *autonomous*, *tele-autonomous*, and *tele-operated* control of robotic manipulators. Using the Telerobot Testbed, we have demonstrated several of the capabilities and technological advances in the control and integration of robotic systems which have been under development at JPL for several years. In particular, the Telerobot Testbed was recently employed to perform a near completely automated, end-to-end, satellite grapple and repair sequence. The task of integrating existing as well as new concepts in robot control into the Telerobot Testbed has been a very difficult and timely one. Now that we have completed our first major milestone (i.e., the end-to-end demonstration) it is important to reflect back upon our experiences and to collect the knowledge that has been gained so that improvements can be made to our existing system. It is also believed that our experiences are of value to the others in the robotics community. Therefore, the primary objective of this paper will be to use the Telerobot Testbed as a case study to identify *real* problems and technological gaps which exist in the areas of robotics and in particular systems integration. Such problems have surely hindered the development of what could be reasonably called an *intelligent* robot. In addition to identifying such problems, we will also briefly discuss what approaches have been taken to resolve them or, in several cases, to circumvent them until better approaches can be developed.

## 1 Introduction

JPL's Telerobot Testbed [7] is an integrated robotic testbed used to develop, implement, and evaluate the performance of advanced concepts in *autonomous*, *tele-autonomous*, and *tele-operated* control of robotic manipulators. The Telerobot Testbed consists of two Puma 560 robots mounted side-by-side; one Puma 560 mounted adjacent to the first two, called the Vision Arm; a taskboard situated between the first two Pumas, and a mockup of the Solar Max satellite mounted to a counterbalanced suspension system. The Telerobot Testbed has a number of sensing capabilities, including Force/Torque Sensing on the two Puma 560 robots, a five camera vision system which utilizes custom designed image detection and feature extraction hardware, and end-effector contact sensing. The software which controls the Telerobot Testbed is distributed among a number of computers including several MicroVAX workstations, a Symbolics workstation, and a variety of low level robot servo controllers and sensor system controllers. Each workstation contains the control software corresponding to one of the subsystems within the Telerobot Testbed's hierarchical control structure. The hardware and software integration of all these components has been a major task. The Telerobot Testbed provides a unique environment for developing integrated and intelligent robotic capabilities.

Using the Telerobot Testbed, we have demonstrated a number of capabilities and technological advances in the control and integration of robotic systems which have been under development at JPL for several years. In particular, the Telerobot Testbed was recently employed to perform a near completely automated, end-to-end, satellite grapple and repair sequence. The task of integrating existing as well as new concepts in robot control into the Telerobot Testbed has been a very difficult and timely one. Now that we have completed our first major milestone (i.e., the end-to-end demonstration) it is important to reflect back

upon our experiences and to collect the knowledge that has been gained so that improvements can be made. It is also believed that our experiences are of value to the others in the robotics community. Therefore, the primary objective of this paper is to use the Telerobot Testbed as a case study to identify *real*, problems and technological gaps which exist in the areas of robotics and systems integration. In the following paragraph we briefly describe one of the basic tasks performed by the telerobot during our previous demonstration in order to place our discussions into context.

The sample task executed by the telerobot system consisted of the removing of a bolt located behind a latched door using a standard socket tool. Task execution was complicated by the presence of a crank in an adjacent area that prevented the door from opening fully for certain orientations of the rank. The task sequence consisted of unlatching the door; using the stereo vision system to determine the orientation of the crank; combining the sensor information with a-priori information to get the best estimate of crank position/orientation; planning and executing a compliant motion to grasp and subsequently move the crank to a non-interfering location; opening the door; and finally holding the door open with one manipulator while the other manipulator arm applied the tool on the bolt. During the actual door opening operation, cooperation of both arms was needed to get the door to open. This was because a single one-arm motion to fully open the door was not possible because of a conflict between joint stop violations and compliant motion stability near the wrist singularity. As a result, the grasp of the end-effector on the door handle had to be adjusted halfway through the opening operation with the other end-effector holding stationary the partially opened door. It should be pointed out that this fixturing operation requiring the cooperation of the two arms was not anticipated during the initial development of the task sequence. Teleoperation and handoff to and from autonomy was also demonstrated for some of the free-motion segments of the task using a run-time path planner in the RTC subsystem.

This paper is divided into three sections corresponding to the three main areas in which have identified problems which must be addressed during the design and implementation of a large integrated telerobotic system. Section 2 describes problems which have plagued our system in the area of calibration and world modeling. Then, in Section 3, we switch to a discussion of the issues relating to process planning and control execution. Finally, in Section 4, we present our ideas as to what features are really important in the design of an overall system architecture and describe how the use of certain approaches to software engineering can, in practice, determine system performance. Section 5 summarizes our conclusions.

## 2 Calibration and World Modeling

Over the past several years and most recently during our efforts to demonstrate the autonomous and teleoperated capabilities of the JPL Telerobot Testbed system we have become acutely aware of the difficulties involved in calibrating an integrated multisensory, multiactuated robotic system. Furthermore, we have learned first hand how great of an impact calibration issues and problems can have upon the design, implementation, and performance of high level control and task planning algorithms. The difficulties that have been encountered in merging information gathered from both vision and touch sensors (i.e., wrist force/torque sensors) in order to estimate the location of an object in the environment are a prime example.

Until recently [2,6,8], the topics of sensor fusion and calibration have received little attention within the robotics research community in comparison to say artificial intelligence for task planning. In response, this section will: (1) highlight several of the problems which we have had to deal with in the design of the Run-Time Controller (RTC), Manipulator Control Mechanization (MCM), and Sensing and Perception (S&P) portions of the JPL Telerobot Testbed; (2) attempt to reinforce the need for increased research in the areas of multisensor system calibration, sensor fusion, and system architectures which explicitly take into account calibration and data fusion. Our experience indicates that the importance of many of the practical issues involved in multisensor calibration and sensor fusion to the design of useful and realizable robotic systems for space applications as well as other applications should not be underestimated.

The autonomous manipulation of an object by a manipulator requires that the location (i.e., position and orientation) of the object be known to within a specified tolerance with respect to the manipulator. In general, this tolerance is dictated by: (1) the positioning accuracy of the manipulator; (2) the accuracy of the object model database which describes the relative spatial relationships between all of the objects in the environment; and, (3) the constraints imposed by the physical configuration and/or motion limitations of the end-effector (gripper). To overcome these inherent inaccuracies, to accommodate mechanical constraints, and to increase system robustness the Telerobot Testbed utilizes contact sensing and stereo vision to assist in the run-time and real-time identification and verification of object locations.

Within this context consider the task of grasping one of the task board's door handles in preparation for opening the door. The door handles are rectangles 2.0 cm by 2.0 cm by 4.0 cm high. This apparently simple task is complicated by fact that the parallel jaw gripper being used has a maximum finger separation of only 2.54 cm. This provides a maximum clearance of only 2.2 mm per side assuming perfect end-effector orientation. This is further complicated by the Puma robot's inability to position its end-effector that accurately, in absolute coordinates, when controlled using standard (i.e., nominal) kinematic parameters. Even if the actual kinematics of the manipulator were identified [8] it is extremely difficult if not impractical to determine the relative spatial transformation between the robot's base coordinate frame and the location of the object to this accuracy. As discussed in [8] and [5] manipulator kinematic models are based upon relationships between nonphysical internal coordinate frames and not to coordinate frames which can be physically identified.

The structure of the Telerobot's global world model database, which drives the RTC planning and event monitoring software, also has a bearing upon this problem. This tree topology connected database provides an on-line object oriented means for computed object positions. The tree topology was based on the inherent physical connectivity of objects, with parent object motion resulting in automatic update of all successor objects. In the case of objects with more than one parent, the spanning tree corresponding to the relational graph was used. Completely independent of the connectivity tree is a measurement tree reflecting the metrology tree used in the building of the initial database. For practical reasons, only portions of this measurement tree coincide with the connectivity tree.

Geometric errors in the world model database are primarily a result of errors in the orientation and position of objects described with respect to other objects. The relative spatial transformations between these objects was initially derived from a combination of mechanical drawing specifications and manually taken measurements. Of these errors, orientation errors were especially troublesome because of the long *lever-arm* effects over workspace distances. With this modeling strategy errors in each of the relative transformations can accumulate into significant absolute location errors. In the case of grasping the door handle, the relationship between the manipulators base coordinate frame and the door handle's location depends upon the relative spatial transformations of between approximately 10 - 20 different primitive objects. The net result has been that when the robot is commanded to move to a cartesian location relative to the door handle, such as a desired grasp approach point, it will often be in error by 5-10 millimeters. Naturally, any subsequent motion to grasp the object will fail unless additional sensory information is used to either update the object database or guide the motion of the end-effector.

An approach which was originally pursued to overcome this problem utilizes a stereo vision system to identify the location of the door handle (or object) and update the database just prior to manipulation. The use of vision, however, has given rise to difficult calibration issues similar to those involved in determining the locations of robot base coordinate frames. In particular, one needs to accurately determine the relationships between: (1) the location of the door handle relative to the camera image frame; (2) the location of the camera image frame and the last link of the robot to which the cameras are attached (i.e., the camera robot); and, (3) the location of the base frame of the camera robot and the base frame of the robot which will perform the manipulation.

The accuracy of first of these relationships is directly related to the accuracy with which one can determine the spatial relationship between the two camera image frames. Comparisons of the relative

locations of object features (e.g., corners, edges) of a known object to feature locations computed based upon a camera model have been used to identify the camera model parameters. In practice, camera model parameter identification is performed off-line prior to system operation. The second relationship, the location of the camera frame and the last link of the robot, is especially difficult to measure since it has an unknown relationship to any of the physically accessible surfaces of the cameras. In fact, the camera frame may likely be located at a point between the cameras in free space. In the testbed, the camera robot is mounted to a wooden table which is bolted to the concrete floor about a meter from the lathebeds which support the two other robots and the task board. Consequently, the camera arm and the other arms have no common physical means for registration. Nevertheless, manual measurements have been used to approximate the spatial relationships between the robots. During the measurement process the orientational deviations between the plane of the camera robot's table and that of the lathebeds are ignored due to the limitations of the measurement equipment. Such deviations, however, can propagate into significant positioning errors between the robot base frames and the frames associated with the objects of the task board.

Although the above approach to object location verification has many drawbacks, it has been successfully used to determine the locations of a variety of objects. In a tele-autonomous system such as ours, the sensed object location information must be incorporated into the system's world model. This process of updating the spatial relationships within the database presents additional problems. The most significant of these is that of maintaining database consistency. Consistency in update is maintained when geometric relationships between objects are not violated as a result of the update. For instance, we know that the door handle does not intersect with the door; rather they share a common surface. Similar statements can be made about the door, its parent object, and so on. But the vision system, due to calibration errors as well as measurement errors will likely return an answer which would, according to the database, place the door handle partially within the door. While systematic approaches to solving this problem have been proposed [6] they are generally unsuitable for implementation due to their large computational requirements and/or their need for statistical parameterizations which are unavailable. Maintaining such consistency in the absence of a generalized approach proved to be extremely troublesome and hence efforts were initiated to provide an algorithmic approach to the problem.

In terms of database philosophy, geometric uncertainty representation per-se is not critical. Our planning methods did not reason with uncertainty and hence it was only necessary to project all of the geometric uncertainty onto a unique and *certain* geometric world model state. Similarly, regarding reasoning with uncertainty, an approach that updates the world-model after each execution step with the best estimate of the geometric state seems to be adequate. Propagating uncertainties to future actions may not be a profitable approach given the paucity of techniques for modeling such uncertainties.

Meanwhile, in order to satisfy various short term needs, we have applied several ad hoc schemes to manipulate and update the world model spatial relationships. These schemes use a concept called *localized models* whereby the a priori relative spatial relationships between objects within various portions of the database (i.e., subtrees) are assumed to be perfect. If the location of an object within a subtree is sensed the required change in the object's location is propagated to an equivalent change in the spatial relationship between the parent object of the subtree and its parent.

This concept is now being applied in conjunction with vision based object verification to provide a vision based relative calibration capability. The objective of this calibration is to apply the vision system to view and locate both the object of interest as well as the end-effector, and to determine the relative transformation between the two. Based upon this relative transformation, the end-effector can then be moved to a location that is within the a priori limits required for subsequent task execution. This approach takes advantage of the fact that the relative accuracy of the vision system is significantly greater than its absolute accuracy. The information obtained from the vision system will then be integrated into the system's world model using the *localized model* approach. In particular, the update is achieved by assuming that the manipulator's kinematics and kinematics used in control are precisely the same. Thus the absolute location of the end-effector is specified by the measured joint angles. The location of the root object of the localized model is then updated such that the object/end-effector relative transformation is identical to that which is

observed. Then, even if the absolute locations of the end-effector and its immediate surroundings are not known with great accuracy one can still have confidence in success of subsequently planned autonomous fine-motions.

A drawback of the *localized model* concept, however, is that a significant amount of bookkeeping is required to maintain a list of the satisfied versus unsatisfied geometrical constraints which will exist within the database. Following long periods of operation it can be argued that the knowledge as to the overall state of the world may actually decrease despite the fact the large quantities of sensory information have been obtained. Implicitly, information is continually being discarded each time the connectivity of localized model is changed. The ad hoc nature of this approach is not well suited for large complex systems since it requires significant amounts of custom engineering. In fact, we have experienced a number of situations in which ensuing side effects have changed the apparent behavior of our path planner and collision detection algorithms. Fortunately, the ever present kill button has saved us from damaging the testbed.

This section has presented a few of the problems which we have faced in the areas of calibration and world modeling. This presentation is by no means exhaustive. We hope, however, that this introduction provides some awareness to the problems being faced by those trying to build real robotic systems and, in particular, robotic systems for space applications. Conclusions are presented in Section 5.

### 3 Process Planning

Planning for telerobot processes i.e. actions such as move, grasp, rotate etc. presented special problems because of the complex and interacting nature of a very large number of problem constraints and characteristics. Among these were the usual constraints arising from the kinematic construction of the Puma 560 arm such as reach, joint stops (especially joint 5), and limited manipulability over large regions of the workspace. Added to these were spatial constraints in the form of inter-arm collision avoidance for the three arms, arm/object collision avoidance, object/object interference avoidance during manipulation, and object occlusion during vision operations. Dynamic constraints included performance degradation and even instability of compliant motion operations near singular arm configurations, and poorly known gravity and friction effects.

The effects of these constraints were further compounded by the requirement to maintain fidelity to the telerobot demonstration task. This implied that in order to preserve the resemblance to a space servicing task, the environment could not be significantly simplified (e.g. by performing the operations on a flat clutter free table). Further, inordinate amounts of computing resources could not be expended during the planning process. This was both because of execution time constraints imposed by operator/machine interaction requirements and because machine speed was constrained by the available hardware. While the hardware speed restriction could have been overcome by upgrading the laboratory system, the inavailability of high-speed space-qualified computing hardware would have equally hampered a real space application.

The demonstration task also required that elemental robot actions be concatenated to perform a meaningful task. Concatenation implies that actions selected in the current step affect the constraints on future tasks, and requires the ability to invoke some form of constraint propagating, backtracking planner. Some of the constraint propagations only affected simple geometric aspects of the task (e.g. selecting a grasp point to facilitate a future ungrasp and depart operation) whereas others changed the entire task sequence (e.g. a forced dual-arm coordination to overcome an individual arm's manipulability constraints). The concatenation of actions also placed a heavier burden on verifying the successful completion of each step. Design of recovery procedures in case of off-nominal execution was also necessitated, especially given the highly variable friction effects in metal-to-metal contact, and calibration uncertainty effects in different regions of the workspace.

The final challenge in the area of planning was to successfully blend engineered solutions to algo-

rhythmic approaches. The boundary between the two was often fluid as algorithms which did not work were engineered to do the job and vice-versa, though admittedly more of the former than the latter.

The combination of factors outlined above made a standard rule or logic based planning system very difficult to design. An alternate approach in the same spirit of the *generate and test* paradigm was adopted instead. This approach utilized action selection rules and procedures based on a partial handling of the various constraints to generate candidate actions. Detailed world-model driven simulation of the action was then used to accept or reject the candidate action. For example, the path segment design algorithm focussed on simple single-arm kinematics and collision avoidance features of the task to generate a candidate motion segment. The acceptance of the motion segment was based on performing a detailed high-fidelity model-driven simulation of the segment to unearth manipulability constraints and collision problems with the other arm. From the viewpoint of the planner based on this concept, actual execution results are indistinguishable from the simulated results. Thus collision during the guarded motion execution of the path segment constituted the same information to the planner as a world-model simulation indicated collision along the path segment.

This generate and test cycle of operations owed its success to two features (1) the ability to synthesize candidate actions that have a high probability of success (2) the use of simulation based checking of such candidates. While it is intuitive to use the best possible methods to determine the candidate actions (e.g. a configuration space graph search for paths), the introduction of a performance criteria that penalized execution speed and assigned a weighted score to the probability of success showed that it was sometimes better to utilize approximate but computationally cheap methods to generate and test the candidate actions. For inherently computationally hard problems such as spatial planning in a variable environment where pre-planned paths are inappropriate, such an approach may indeed be the only feasible one [3].

The generate and test paradigm also lends itself to easy operator interaction in the overall decision making. The action choices can be selected and pruned by the operator and the simulation process can use operator input for compensating for operator modeled phenomena not modeled in the planner or for judgment based acceptance of candidate actions. One useful side benefit of the simulation process was the potential ability to plan sensor monitoring processes. For example, during the path segment simulation, objects neighbouring the path segment could be monitored for distance and direction and reflex actions or sensor event detection predicates could be synthesized accordingly. This capability was however not exercised in the laboratory and hence we have no direct assessment of its utility.

## 4 System Architecture and Software Engineering

In a well established and understood field, the construction of a system without first exploring its formulation and design in great detail is foolhardy. Unfortunately, this rigor is not feasible in a research environment, where requirements fluctuate at an extremely high rate. In these situations, success is often determined by the ability to rapidly react to significant design changes. The JPL Telerobot demonstrator project clearly lies within the research domain, and the system design was incrementally modified every six months or so. The size of the system (which eventually exceeded 200,000 lines of software and 5 primary CPU's) and the short development time (less than two years from initial concept to first major integrated demonstration) necessitated an extremely efficient use of time and hardware resources (physical manipulators, not computers). One of the consequences of this was the requirement for interactive development placed on most subsystems and subsystem components. This had a strong influence on the architectural evolution of the system, as we discovered the advantages of a fully interactive environment. In the following we briefly describe the issues and insights that have been encountered in the area of overall system architectural philosophy.

It is essential that all system capabilities be interactively available for use by the human, since augmentation of human capability is much more effective for robust, flexible telerobotics than using purely autonomous robots or pure force-reflecting teleoperation, which is not robust in the presence of time delays.

Our experience indicates that the low- and medium-level actuation and sensing subsystems, which may have been originally designed for use by higher level software, are in themselves a very powerful set of tools which can be used directly by a human to perform tasks, without the need for any complex planning or analysis software. This observation, that a human can often serve as a functional component in a software system, is significant because a human can usually replace vast quantities of complex software with only minor overhead for provision of appropriate user interfaces to make the human appear to the rest of the system as just another software module. Although this point has implications for software development and debugging, it crucially impacts the design of the overall system architecture, since it can make differences of orders of magnitude in the ability of a large robotic system to perform practical tasks in a realistic environment. Typically, it was relatively straightforward for the developers of the RTC to use their interactive software development environment to control the actuation and sensing subsystems, and the individual algorithms and internal modules of the RTC, *by hand*, enabling them to perform tasks which were much more complex, *and more robustly executed*, than the integrated RTC autonomous software could attempt.

Existing work in telerobot systems architecture [1] has concentrated on a perspective of well-structured environments which allow the design of the telerobot to concentrate mainly on how to divide the task to be done efficiently between several cooperative robots. Our work indicates that space telerobotics is not a highly-structured task, and that concentration should be placed on the reactive, feedback portion of the architecture, which allows the robot to respond to the environment. To this end, the cooperative human-machine concept gives much greater robustness and flexibility than any existing or proposed autonomous system promises.

In most large robotic systems, there is a low level subsystem which functions as a server of basic actuation/sensing functionality to the rest of the system. Typically, it is *not* the case that a higher level system can use its interface to actually command any given motion or sensing operation which the hardware devices are physically capable of performing. There are usually many classes of actions which cannot be commanded due to interface "blind spots". Practical experience indicates that such blind spots often lie precisely in the regions which later turn out to be crucial to the functioning of the system as a whole, and thus must be avoided at all costs. Each low level subsystem must be built from the perspective that anything which its "designer" would ever want to do with it can be done *through the interfaces provided to other subsystems*. Often, this is not the case, since the builders of the low level subsystems often work in a very flexible software development environment during implementation, but do not themselves rely upon the interface to perform their own tests. The ease with which they can directly use their subsystem to perform tasks, owing to the great robustness and adaptability of humans, misleads them into thinking that it will be quite sufficient for use by the higher levels of software. Correction of any design omissions which lie in the blind spots then becomes very costly.

The ability to shift functionality across subsystem interfaces easily is also crucial. During initial design, conceptually complex tasks are often divided up semi-arbitrarily and choices are made for locations within the various subsystems for each function, usually based largely upon such considerations as "level of intelligence", computational and bandwidth requirements. However, since the introduction of a new algorithm with improved functionality can force the shifting of a large body of capability from a higher level subsystem to a lower one, the architecture of the system should conveniently support it.

We now present several issues which we found to be of prime importance during the software development process. They are well-understood by the professional software engineering community, but the small size of most robotics developments to date has not yet provided much cross-fertilization. Our aim here is to convey a healthy respect for the magnitude of the software task for large integrated systems, which can easily overshadow the robotics issues in terms of time and effort spent.

First of all, the most important issue in large systems software development is human, rather than technological: *frequent, detailed communication between virtually all members of the development team is absolutely required*. Top-down management techniques can provide some aid to the development of large software systems *if the top-down method is applied to the system as a whole, rather than just one or more*

*subsystems*, but the bulk of the design effort must be done by the many individuals on the various development teams. To prevent minor choices on the part of subsystems from propagating disastrously to the system, every individual must understand enough of the overall system to measure his decision's impacts on it. This is not an easy situation to maintain, but the alternative is a collection of independently powerful subsystems whose integrated capability is much less than the sum of their individual talents. The next paragraphs discuss some of the issues which specifically relate to the software development process itself.

The techniques which individuals use to manage and design small software systems are often precisely those which create the greatest difficulties in a large system developed by a team [4]. We have learned that there is a great need to impose at least the minimum of standards and techniques from the professional software development regime, in spite of many individuals' resistance to them on the grounds of the additional overhead they impose. In the absence of unusually disciplined developers with an excellent rapport, a system built without careful control of software development will be significantly, if not overwhelmingly, inferior to what it could be, in performance, development time, resources used, and future adaptability. In general, good software discipline can make the difference between a team which is performing research, and a team which is continually struggling so heavily with software and hardware implementation difficulties that they have virtually no time for broad thought about the goals and subtleties of the system they are building. Underestimating the difficulty of a hardware/software item can be one of the most dangerous mistakes to make, because once hardware/software problems pass a certain degree of desperation in some subsystem, the members of that subsystem's development team tend to withdraw from the overall system design effort due to lack of free time, and all the decisions they make from then on will then not only be made with a very narrow and short-term focus, but will not even be communicated to the system as a whole until they are largely irreversible.

Exacerbating the complexity of software development for systems of this size is the fact that systems are built as a pyramid, with low level software tools and low level subsystem interfaces being used as a foundation. Changes to these tools and interfaces which occur without extremely careful monitoring by all members of the system development team can have catastrophic effects, since pyramidal software systems tend to behave like houses built of cards: pull one out of the bottom and everything falls. In the area of performance, it was discovered that the choices which turned out to have the greatest negative performance impact on the overall system were those which were made by individuals working within the subsystems who did not recognize that their decision impacted the system as a whole, and thus did not consider it important enough to bring to the attention of the systems-level group.

Abstraction is probably the single most important software concept in building large systems, precisely because of the pyramidal nature of software. Abstraction, sometimes called implementation-hiding or factoring (which is distinct from modularity), is the practice of inserting additional layers into implementations for the express purpose of decoupling external users from the internal details of a piece of software. A trivial example is the following: suppose a library of graphics routines was rebuilt to use absolute coordinates in centimeters rather than coordinates relative to the display screen. If the library were built as a single layer, there would be no choice but to change every bit of code using the functions to perform the conversion of coordinate systems, which would then also become device-specific for each graphics monitor. However, if an additional level of abstraction had been intentionally inserted into the library when it was built, the conversion could be performed in this intermediate layer, internal to the library, so that the change in implementation need not propagate outside. This obviously imposes some additional overhead, but is almost universally preferable to performing complicated additions and modifications to previously stable software. The larger the system the more certain changes are to occur, the more severe their repercussions can be, and the more important the use of conceptual abstraction is to software design. In particular, it is essential that subsystems use sufficiently abstract interfaces to prevent minor internal modifications from being propagated to other subsystems and the system as a whole.

The need for abstraction is one of the strong drivers behind a philosophy of development which we call *tool-oriented*. The need to insert additional conceptual layers forces the developers to be continually examining at each subcomponent they are building to see how it generalizes, and what sorts of changes



and additions might be likely in the future. This effectively transforms the design of each subcomponent from a very specialized exercise into the task of *designing a set of software tools* which can be used in a very straightforward way to build the desired subcomponent. This tool-oriented methodology imposes additional overheads and larger initial investments of time and resources, but, we have estimated (based on our experiences when we did not pursue this course) that our long-term productivity has been roughly twice as high as it would have been had we initially built exactly what we thought we needed. The reason for the large gains in spite of the high initial investments required is that research systems require a high frequency of changes and it is vastly easier to rebuild something built with modular tools than it is to rebuild a single monolithic program.

Robustness is also an issue in software engineering and usually is far more important to large systems than efficiency. The tightly-coupled software environment of a large system forces every component to do double duty as both a functional unit and as a diagnostic tool for other modules and the system as a whole. The additional investment in time, both for coding and during execution of the additional error-checking, is well worth the added reliability. In a system with tens of developers and several dozen modules, if the low levels are not reliable, the higher level modules will be forced to devote significant portions of their effort to coping with the drawbacks of the modules they depend upon.

In summary, the task of managing software development for a large system is very different from that of a small one, and most of the decisions which can be safely made with only small consideration and little fear of error for small projects become magnified into major issues for large ones. It may, however, be quite difficult to convince those without experience in software engineering of this. The issues discussed above may not appear to be very significant to them, since they may not recognize how significant the gains in performance and robustness would be from their system if short-term choices were sometimes sacrificed to long-term needs.

## 5 Conclusions

In the area of calibration and world modeling we believe that: (1) the difficulty and cost associated with calibrating a multi-actuated, multi-sensor robotic system is grossly underestimated; (2) seemingly small calibration errors in a complex robotic system can easily propagate into large errors and thereby significantly reduce the overall capability and performance of the systems as a whole; (3) low-level subsystem calibration problems can have a significant impact upon the design of higher level subsystems; (3) the calibration of all of the actuation devices (e.g., manipulators and grippers) and sensors (e.g., force/torque, position, vision, etc) which comprise a robotic system must be performed within a coherent and common framework; (4) ad hoc and/or custom engineered methods for maintaining world model consistency, such as the *localized model approach*, are not suitable for large scale robotic systems; and (5) research in robotics needs to focus, in part, on the development of generalized methods for representing, manipulating, and propagating object spatial uncertainties in a language (e.g., geometry) suitable for use by task planning and control algorithms. In short, do not take the effects of calibration errors lightly.

Robot planning for real world applications becomes extremely difficult because of the complex and interacting nature of the various physical constraints and run time uncertainties. The effect of these becomes especially evident if long sequences of robot tasks need to be performed in a robust manner. In this paper, some of these challenges have been outlined and approaches such as fixturing and world model simulation have been discussed. A planning and execution method based upon a probabilistic approach incorporated into a *generate and test* paradigm has been advocated for its efficacy in overcoming computational constraints associated with these types of problems.

In the area of system architecture, the key features we believe are important are: (1) the interactive cooperation between human and machine during operation of the system to enable the human and the autonomous system to call upon each other as tools to perform specialized functions; (2) the importance of

the reactive feedback loop as opposed to nominal execution of preplanned tasks; (3) insuring that there is no functionality lost between the level of the sensing and actuation and the level of the operator. In the area of software engineering, we found the important issues to be: (1) interpersonal communication among the development team members; (2) discipline during the software development process; (3) high levels of conceptual abstraction; (4) build tools with which to build applications, not single applications; (5) emphasize robustness over efficiency; and, (6) interactive rather than incremental development environments.

## References

- [1] J. S. Albus, H. G. McCain, and R. Lumia. *NASA/NBS Standard Reference Model for Telerobot Control System Architecture (NASREM)*. Technical Report , National Bureau of Standards, Robot Systems Division, National Bureau of Standards, Robot Systems Division, December 1986.
- [2] P. K. Allen. *Robotic Object Recognition Using Vision and Touch*. Kluwer Academic Publishers, Boston, MA, 1987.
- [3] J. Balaram. Probabilistic methods for robot motion determination. In *SPIE Symposium, Advances in Intelligent Robotic Systems*, Cambridge, MA, November 1988.
- [4] F. P. Jr. Brooks. *The Mythical Man-Month (Essays on Software Engineering)*. Addison-Wesley, New York, 1982.
- [5] M. Driels, B. Mooring, Z. Roth, and L. Everett. Fundamentals of manipulator calibration. In *the Proceedings of the 1988 IEEE International Conference on Robotics and Automation*, Philadelphia, PA, April 1988.
- [6] H. F. Durrant-Whyte. *Integration, Coordination and Control of Multi-Sensor Robotic Systems*. Kluwer Academic Publishers, Boston, MA, 1988.
- [7] P. S. Schenker. NASA research and development for space telerobotics. *IEEE Transactions on Aerospace and Electronic Systems*, 24(5):523-534, September 1988.
- [8] H. W. Stone. *Kinematic Modeling, Identification, and Control of Robotic Manipulators*. Kluwer Academic Publishers, Boston, MA, 1987.

## THE KALI MULTI-ARM ROBOT PROGRAMMING AND CONTROL ENVIRONMENT

Paul Backes<sup>†</sup>, Samad Hayati<sup>†</sup>, Vincent Hayward<sup>‡</sup>, and Kam Tso<sup>†</sup>

<sup>†</sup>Jet Propulsion Laboratory  
California Institute of Technology  
Pasadena, California

<sup>‡</sup>McGill Research Center for Intelligent Machines  
Montréal, Québec Canada

### Abstract

The KALI distributed robot programming and control environment is described within the context of its use in the JPL telerobot project. The purpose of KALI is to provide a flexible robot programming and control environment for coordinated multi-arm robots. Flexibility, both in hardware configuration and software, is desired so that it can be easily modified to test various concepts in robot programming and control, e.g., multi-arm control, force control, sensor integration, teleoperation, and shared control. In the programming environment, user programs written in the C programming language describe trajectories for multiple coordinated manipulators with the aid of KALI function libraries. A system of multiple coordinated manipulators is considered within the programming environment as one motion system. The user plans the trajectory of one controlled Cartesian frame associated with a motion system and describes the positions of the manipulators with respect to that frame. Smooth Cartesian trajectories are achieved through a blending of successive path segments. The manipulator and load dynamics are considered during trajectory generation so that given interface force limits are not exceeded.

### I Introduction

Research and evaluation of robotics concepts are often hindered by difficulty in the implementation process. Current robot programming and control environments have limited task description and implementation capabilities or have hardware/software environments which are difficult to modify, thus making them difficult to use in quickly changing implementation environments. A wide array of research efforts in robotics are presently being pursued at JPL. The present implementation environment, while quite powerful, is difficult to modify to include the added capabilities that the research efforts are producing. This situation has prompted a collaborative effort between JPL and McGill University to create a new robot programming and control environment, Kali, which has simple but powerful task description capabilities and an open modular architecture to provide for evolving capabilities.

A basic goal of the robotics research at JPL is the development of a space telerobot system. The JPL telerobot testbed is used to develop, test, evaluate, and demonstrate state of the art space technology through development of likely space robotics tasks such as satellite grappling and repair and orbital replacement unit changeout. Several subsystems, from

low level task execution through sensing, vision, and planning, coordinate efforts to execute the tasks. The Kali system will be used for the low level task execution and sensing part of the JPL telerobot system. At this level many research areas demand a powerful, flexible system for implementation. These research areas include shared control, traded control, multiple arm control, redundant arm control, teleoperation, task space description and control, force/position hybrid control, sensor design and integration, adaptive control methods, flexible arm control, and human factors.

The present robot control environment used for JPL telerobot research is Multi-Arm RCCL which provides for coordinated motion of two arms [1]. This is an extension of the original version of RCCL developed by Hayward at Purdue University [2]. Kali utilizes many of the concepts used in RCCL. Results of work on Kali prior to this paper can be found in [3,4,5,6,7].

## II Standard Kali

Kali is a low level software and hardware environment for trajectory description and control of multiple coordinated machines, e.g., manipulators, walking machines, or robotic hands. The initial Kali system will be implemented specifically for control of multiple coordinated manipulators. At the high level of Kali, a collection of C language functions are provided which allow the user to describe and implement coordinated motion of multiple manipulators using a C language program. The lower level of Kali has processes distributed over multiple processors in a VMEbus environment.

It is desired with the user interface to Kali to provide the user with an environment to, as easily and generally as possible, describe and execute a task which requires the motion of one or multiple coordinated arms. This is done by separating the task description from the specifics of the underlying mechanical system. The user describes the task in task space, i.e., the motion and forces of a Cartesian frame along with task space constraints. Underlying mechanical system constraints can be specified independently from a specific task and can be considered automatically by Kali which removes the burden of the details of the underlying mechanical system from the user.

Dynamics constraints of the underlying mechanical system are considered in trajectory generation. The load limitations of a manipulator can be specified and a trajectory will be determined which does not require the arm to produce forces greater than the limits specified. This requires the user to provide dynamics models of all manipulators and objects.

An important feature of Kali is its modular functionality. Modularity simplifies software and hardware customization resulting in a convenient environment for research or customization for a specific installation. Software modules such as trajectory generation and servo control have defined functionality allowing them to be replaced without altering other parts of the system.

The hardware environment was selected for simplicity of implementation, computation speed, and upward compatibility with advancing processor technology. Kali utilizes a distributed hardware environment which allows for parallelization of computations, simplified sensor integration, and modular hardware functionality.

## II.1 Motion Systems

Motion description in Kali is specified in terms of a motion system. A motion system describes the motion of a Cartesian coordinate frame, the Control frame, and the constraints associated with the Cartesian motion. User task description involves describing the motion of the Control frame in terms of time synchronization, destinations, velocities, and transmitted forces. Coordinated motion of multiple manipulators is achieved by kinematically constraining the manipulators to the Control frame. Multiple motion systems can be described and executed allowing manipulators to operate independently or semi-coordinated by specifying related completion times.

All information describing the motion of a motion system is placed in a motion record and motion records are placed in a queue to be processed on a first-in first-out basis by the trajectory generator. Motions are processed by a finite state model where the motions go through a sequence of states depending on control flags.

## II.2 Spatial Relationships

Spatial relationships in a motion system are described by frame transformation graphs in the form of ring structures. Each ring structure, or loop, in the structure is equivalent to the equation

$$B M T D C = Identity \quad (1)$$

where B represents the manipulator base transform from a fixed world frame to the manipulator base, M represents the manipulator transform, T the tool transform from the manipulator final link to the controlled frame, C the goal position of the control frame, and D the drive transform which is interpolated from an initial value which satisfies the initial state to the identity transform in order to produce the desired motion. Each transformation ring in a motion system is set up by one position making primitive. All loops of a motion system share a common drive transform. Spatially coordinated motion of manipulators which are not part of the same motion system is possible by having common transforms in their ring equations other than the drive transform, e.g., common control frame goal positions. The ring equations allow for additional frames in the loop such as for sensor based motion as described below.

Each transform in the loop has associated with it a bound function which is used to update the transform each sample interval. Standard functions such as those bound to the M and D transforms are supplied by Kali, although the user can specify new ones simply by placing pointers to these in the ring equation. Other functions such as those bound to the A transforms of figure 2 below are supplied by the user to provide for specialized motion such as sensor based motion for compliant control. The order of evaluation of the transform-bound functions is specifiable by the user. The normal order is sensor-based functions (e.g., bound to A above), path planning functions (bound to D above), and lastly, update of the manipulator transform M.

Many types of multiple manipulator motion are possible using Kali. Three examples utilizing one and two motion systems are depicted in figures 1 - 4. In figure 1 the two arms share a common motion system to move an object that both arms are grasping. The figure shows the transformations from the ring equation between coordinate frames which are

attached to physical objects. Figures 2 - 4 show only the frame transformation graphs. Figure 2 shows the transformation graph for figure 1 with the addition of the A transforms. The small perturbations caused by the A transforms would be difficult to show in figure 1 which shows the physical objects. Figure 3 depicts the case where two manipulators are moving to grasp a common object. Since the arms have different distances to move, independent motion systems (drive transforms) are used to move the arms. The arms share a common control frame goal position which specifies the grasp points on the object for the arms. Concatenated manipulators using two motion systems such as for a gross-positioning arm with an attached micromanipulator are shown in figure 4.

### II.3 Dynamics

The trajectory generator automatically considers the dynamic constraints of objects associated with frames in the ring equations when determining the parameters of the motion system drive transform. Functions describing the dynamics of the objects, including the manipulators, must be provided by the user. These functions return acceleration, gravity, and velocity terms. Standard dynamic model functions, such as for solid cubes, will be provided by Kali. The user may also specify the maximum force that can be taken at an interface. For example, a manipulator's load limit can be specified in this way and the trajectory generator will produce trajectories which will not violate this limit.

### II.4 Trajectory Specification

Trajectories in Kali are considered as a string of Cartesian linear path segments connected by transitions. Velocity is controlled along a path segment and acceleration is controlled during transitions between path segments. The Cartesian trajectory function generates the motion parameters of a drive transform for a motion system such that the specified spatial, temporal, and dynamic constraints are satisfied. Constraints at the task level are described as part of the motion system by the user. The Kali-supplied trajectory generator is described in [5].

The transition between segments is accomplished by the blending of successive path segments with blending controlled by preview and acceleration factors and accelerations limited by dynamic constraints. The preview factor conveys the amount of look ahead the system must perform before a transition. For example, for a preview factor range of 0 - 1, a value of 0 indicates that the trajectory generator has no knowledge of a new path segment ahead and stays on the current path segment through the goal position. At this time, the start of the next path segment, the transition begins, thus causing an overshoot. For a preview factor of 0.5, trajectory wander off of each of the successive segments is equally permissible during the transition. The acceleration factor conveys the amount of admissible trajectory wander. A small acceleration factor specifies a smaller admissible wander, thus causing a longer transition time. This blending method is robust to ill-defined trajectories since it does not rely on position, velocity, and acceleration boundary conditions.

Other motion system constraints include beginning and ending times, coupling factors, and maximum transmitted forces. Beginning and ending times can be used to synchronize motions of different motion systems. The amount of force transmission between each kinematic relationship is specified with a coupling factor. This factor is normally set to 1, which means

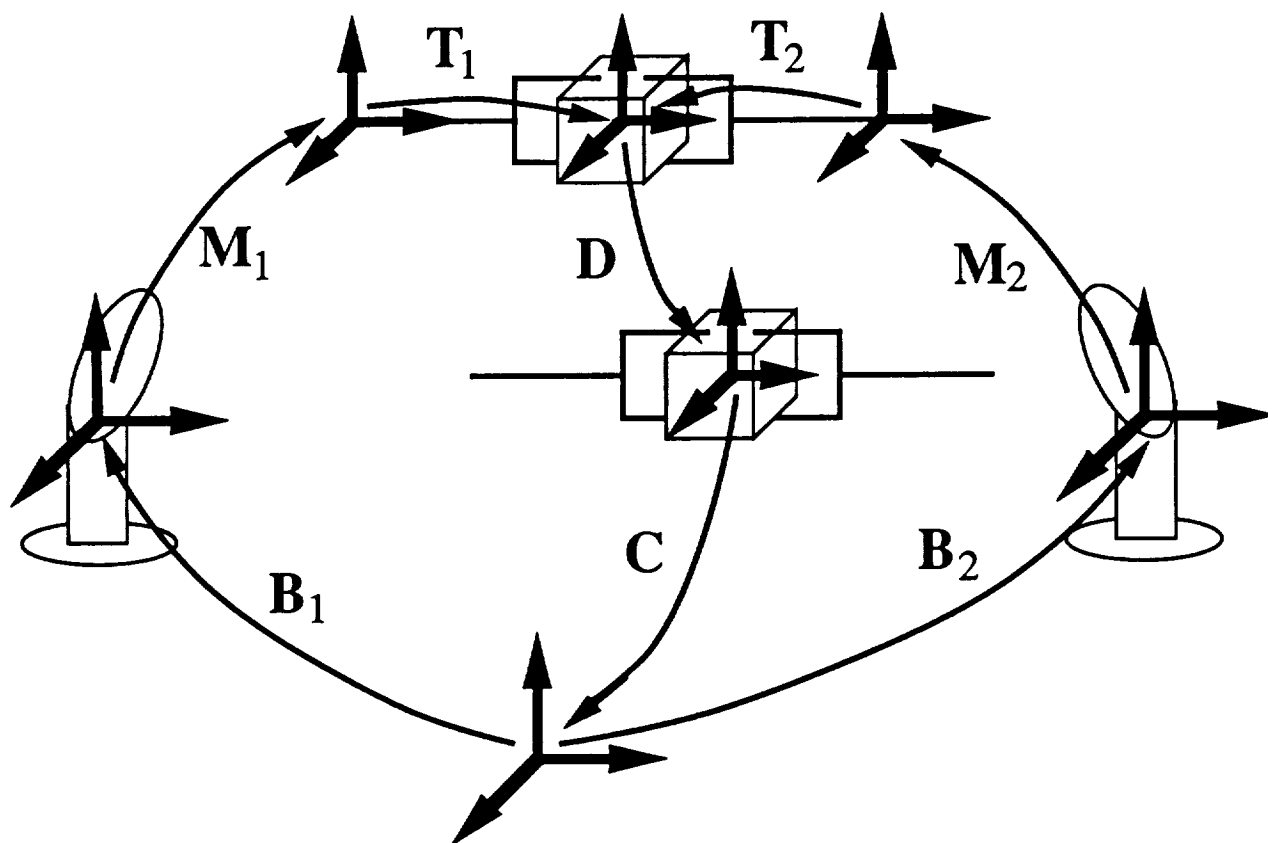


Figure 1: Two arms manipulating a common object

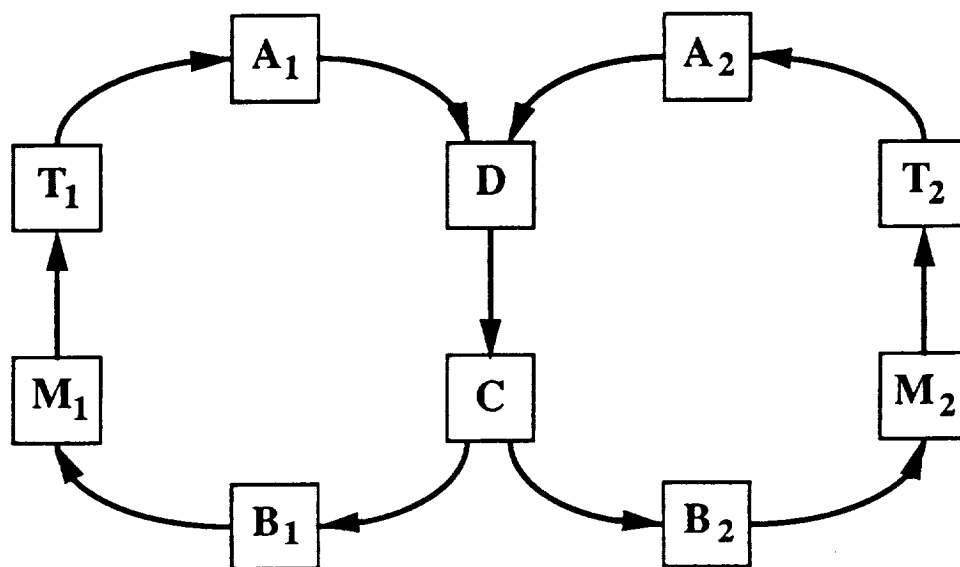


Figure 2: Transformation graph for two arms manipulating a common object

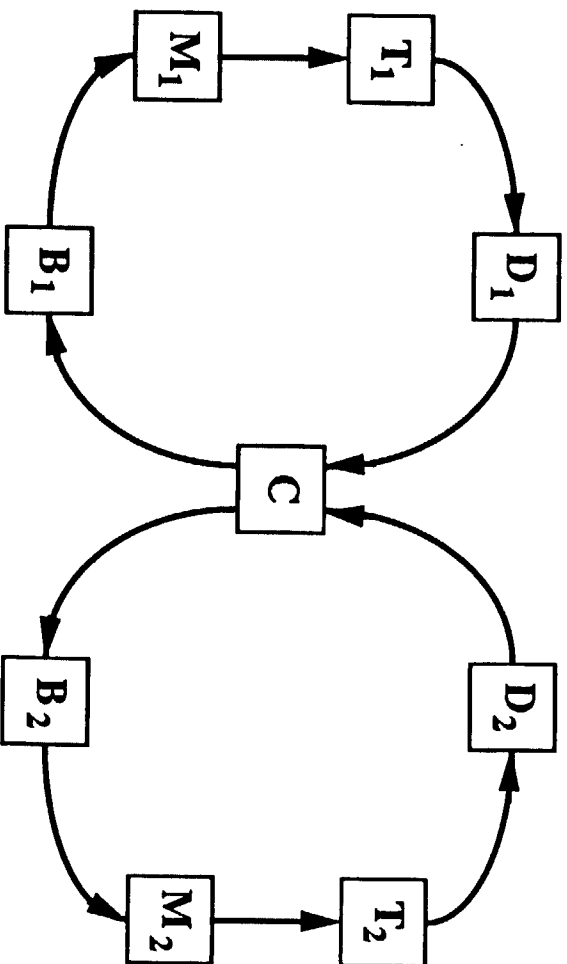


Figure 3: Two arms moving to grasp a common object

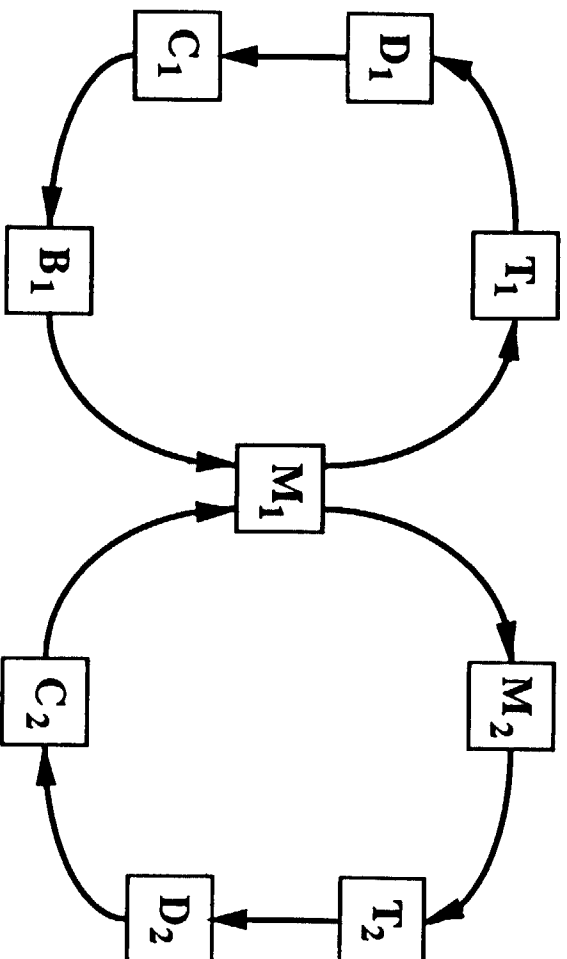


Figure 4: Micro-manipulator concatenated at terminal link of gross-positioning arm



that all forces applied to an object attached to that relationship must be accounted for by the next object. The coupling factor can also be used for load sharing in multiple-arm control. Coupling factors of 0.5 between the load and each of the two arms can be specified indicating that the trajectory generator considers each arm to dynamically carry only half of the load, thus allowing increased speed versus a one-arm motion. The maximum force that can be transmitted at an interface can also be specified. The transition time is initially determined from the temporal constraints and increased as needed to satisfy the dynamic constraints specified by the maximum transmitted forces and determined by the dynamics functions and the coupling factors.

## **II.5 Control**

Kali does not provide a standard servo level control algorithm for controlling the manipulators. Instead, Kali provides setpoints for the servo control and the interfaces to the control algorithms. Several control schemes for possible use within Kali are presently under study [6].

## **II.6 Software**

The Kali software environment is written in the C programming language and organized into five layers, each of which contains multiple libraries. The first layer, MUX (McGill University Extensions) runs on top of the commercial real time multi-processor operating system VxWorks (from Wind Rivers Systems, Inc). MUX provides an environment for running various synchronous and asynchronous processes on multiple processors [7]. The second software layer is also a support layer but is independent of the operating system. There are libraries for buffered input and output of data, geometric computations on vectors, transformations, quaternions, and kinematics and dynamics models of the manipulators. The third software layer implements the servo control using the lower level software layers. The fourth layer updates the kinematic loops in real time using two libraries. The cmotion library is used to compute Cartesian trajectories given the constraints in the motion records. The rings library maintains and updates the kinematic loops. The fifth layer continues to be developed. It consists of user level functions to simplify task description, e.g., accommodation functions and functions specific to dual arm control.

## **II.7 Run Time Processing**

Synchronous and asynchronous run time processes are distributed over multiple processors on a VME backplane to increase speed through parallelization. Bus bandwidth is reduced by placing separate functionalities on separate processors. The resulting processor allocation method is simple rather than optimal to facilitate implementation and evolution. Multiple update rates are utilized so that rapidly changing quantities (e.g., position, sensed forces) are updated more often than slowly changing ones (e.g., inertia characteristics, Jacobian). Asynchronous processes may include the user process, dynamics computations, and computation of the Jacobian. The user process initializes the Kali environment, describes the motion systems, issues motion requests, and communicates with higher level systems. Synchronous processes include the Cartesian setpoint generator, servo control, and sensor I/O. Communication between processes is done with message passing and shared memory. An example allocation of processes may be having the user process and trajectory generator on CPU 1, dynamics on

CPU 2 and CPU 3, kinematics on CPU 4, and servo control on CPU 5 and CPU 6.

## **II.8 Hardware Environment**

The Kali hardware environment consists of multiple processors connected on a VME backplane. Presently Heurikon single board computers with the Motorola MC68020 and MC68881 are used at 20 MHz with 1 Mbyte of RAM. A secondary VSBbus serves to access shared memory for all asynchronous communications. C language software is developed and compiled on a SUN workstation under Unix and downloaded to the processors on the VMEbus via Ethernet. Use of the VxWorks real time operating system provides for processor upgrade (e.g., MC68030, SPARC) as processor technology improves. Presently an initial version of Kali is running at McGill University with 1 KHz sample rate.

## **III JPL Implementation for Telerobot**

As explained in section I above, one of the basic goals of robotics research at JPL is the development of a space telerobot system. The standard mode of control for the telerobot is shared control [8] where the operator and autonomous control system both have inputs to the system. The modular architecture of Kali will be utilized to incorporate the shared control mode into the system. A module which may be fully autonomous in standard Kali may consist of an interface submodule which merges inputs from autonomous and operator submodules which each perform the functionality specified for the module, e.g., trajectory generation.

The hardware incorporated into the system includes two manipulators (eventually to be a 17 degree of freedom dual-arm torso system from Robotics Research Corp.), servoed grippers, multiple cameras for vision, other sensors (e.g., force/torque wrist sensors), and two force reflecting hand controllers. Several subsystems are involved in task execution including task planning, sensing, and control. Kali will be used for the low level task execution, control, and sensing. An initial proposed hardware setup for the JPL Kali system is shown in figure 5. Kali runs in one VME chassis and communicates with multiple devices. For the two arm system, Kali processes for the separate arms (e.g., kinematics, dynamics, sensing) are put on separate processors. The user program and trajectory generator for both arms are located on a single processor. Actuator commands are sent to robot motor controllers to control the arms and sensor data from the arms is sent back via parallel ports. A VME chassis is used to control and sense each of the two hand controllers. A third manipulator equipped with vision cameras is controlled independently from Kali in the initial system.

## **IV Kali Extensions**

Kali's modular architecture provides for further evolution of the Kali environment. Evolution may be in the form of enhancements of the functions presently in Kali or additions of functionality. For example, the user may be provided with the option of selecting the servo control algorithm to be used, e.g., joint PID, operational space, multi-arm. An environment contact model could be supplied either by Kali or by the user. A world geometric model database could be added to simplify task description. Real-time collision avoidance of both the arms and environment could be included. A more flexible processor allocation scheme could be incorporated — either specifiable by the user or an automatic optimizing scheme. Also, force trajectory generation at the user level could be provided.

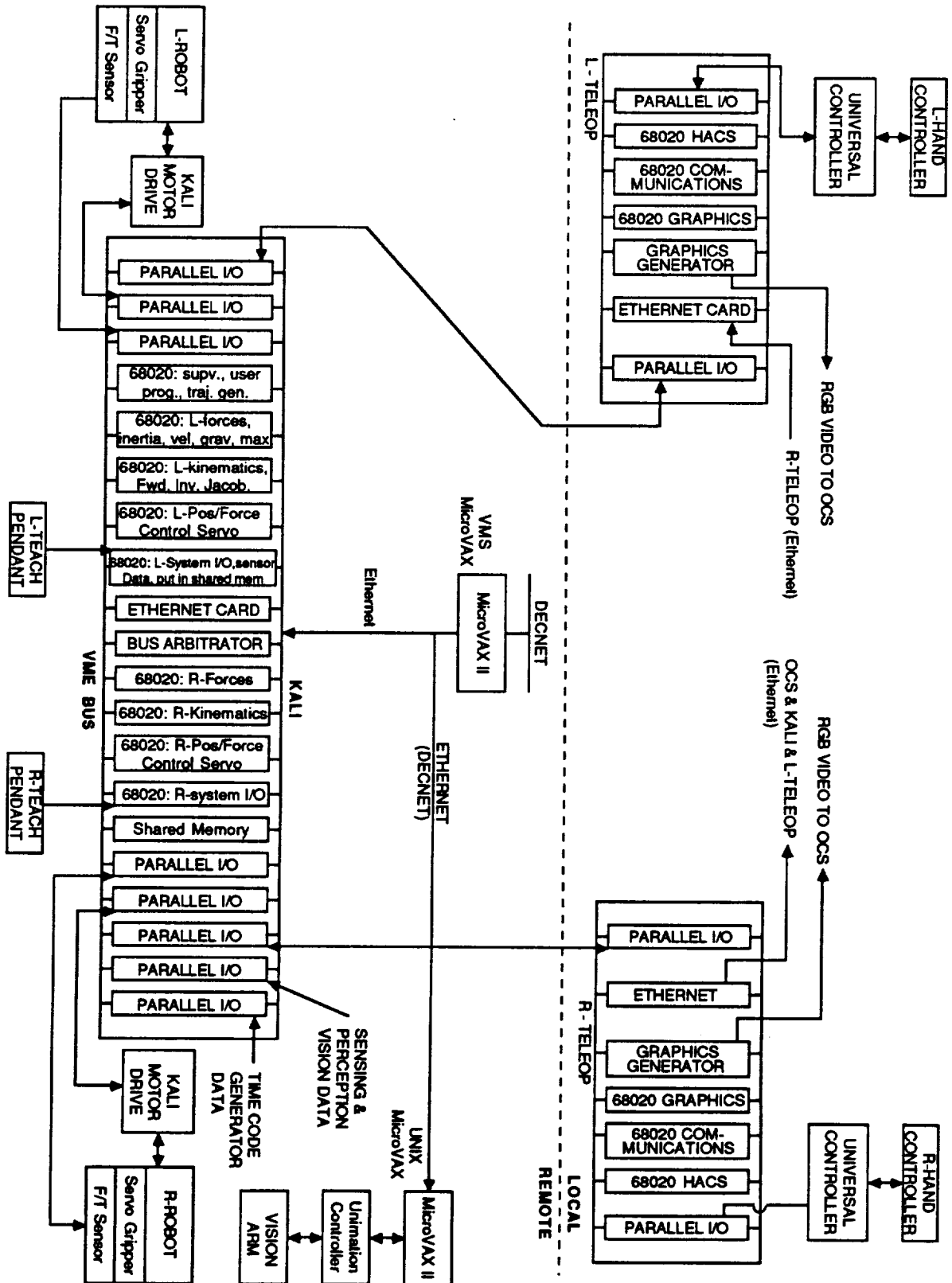


Figure 5: Proposed Kali configuration for low level of JPL Telerobot

## V Conclusions

The Kali multi-arm robot programming and control environment has been described in the context of its use in the low level of the JPL telerobot project. Its advanced features, e.g., multi-arm coordination, dynamically constrained Cartesian trajectory generation, and its open, modular architecture, make it a valuable tool for implementation of advanced robotics concepts. The initial version of Kali, presently running at McGill University, will be further enhanced and ported to JPL's telerobotics laboratory to serve as the low level of the telerobot shared control environment. Taking advantage of its modular design, the Kali environment shall continue to be enhanced as research in the various functional areas of Kali develops.

## Acknowledgements

The research described in this paper was performed at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration. Graphic arts by Donna Milton are also appreciated.

## References

- [1] J. Lloyd, M. Parker, and R. McClain. Extending the rccl programming environment to multiple robots and processors. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 465-474, 1988.
- [2] V. Hayward and R. Paul. Robot manipulator control under unix rccl: a robot control "c" library. *International Journal of Robotics Research*, 5(4):94-111, Winter 1986.
- [3] A. Nilakantan and V. Hayward. Synchronizing multiple manipulators. In *Second International Symposium on Robotics and Manufacturing, Research, Education and Applications*, 1988.
- [4] V. Hayward and S. Hayati. Kali: an environment for the programming and control of cooperative manipulators. In *Proceedings American Control Conference*, 1988.
- [5] V. Hayward, L. Daneshmend, and A. Nilakantan. Model based trajectory planning using preview. In *SPIE Conference, Space Automation IV*, 1988.
- [6] V. Hayward, L. Daneshmend, and S. Hayati. An overview of kali: a system to program and control cooperative manipulators. In *Fourth International Conference on Advanced Robotics*, 1988.
- [7] A. Topper, L. Daneshmend, and V. Hayward. A computing architecture for a multiple robot controller for space applications (kali project). In *Fifth CASI Conference on Astronautics*, 1988.
- [8] S. Hayati and S. Venkataraman. Design and implementation of a robot control system with traded and shared control capability. In *IEEE International Conference on Robotics and Automation*, 1989.

## **TELEROBOT PERCEPTION**



# How Do Robots Take Two Parts Apart?

Ruzena K. Bajcsy and Constantine J. Tsikos

Department of Computer and Information Science  
University of Pennsylvania  
200 South 33rd Street  
Philadelphia, PA 19104-6389

January 31, 1989

## Abstract

This research is a natural progression of our efforts which begun with the introduction of a new research paradigm in Machine Perception, called Active Perception. There we have stated that Active Perception is a problem of intelligent control strategies applied to data acquisition processes which will depend on the current state of the data interpretation, including recognition. In this paper we treat the disassembly/ assembly problem as an Active Perception problem, and we present a method for autonomous disassembly based on this framework.

## 1 Introduction

Perceptual activity is exploratory, probing, searching [1], [2]. Percepts do not simply fall onto sensors as rain falls onto the ground. We do not just see, we look. And in the course of looking, our pupils adjust to the level of illumination, our eyes bring the world into sharp focus, our eyes converge or diverge, we move our heads or change our position to get a better view of something, and sometimes we even put on spectacles.

For robotic systems, this Active Perception approach has several consequences:

1. If one allows more than one measurement to be taken, then one must consider how they should be combined. This is the multi-sensory integration problem.
2. If one accepts that perceptual activity is probing and searching, then data evaluation techniques must be used to measure how well the system is accomplishing its perceptual task and to determine whether a feedback mechanism is needed.

3. If one accepts that perceptual activity is exploratory, then one must determine what must be built into the system in order to perform the exploration, i.e., what is a priori and what is data driven?

The next development in our program was the realization that perception is not only sensing but also involves manipulation [4]. For example, consider the problem of a static scene segmentation. This has been shown convincingly in our recent work [13] and in the paper: "Segmentation via Manipulation" [14] where we argued that a static scene that contains more than one object/part most of the time cannot be segmented only by vision or in general by any non contact sensing. Exception to this is only the case when the objects/parts are physically separated so that the noncontact sensor can measure this separation or one knows a great deal of a priori knowledge about the objects (their geometry, material, etc.). We assume no such knowledge is available. Instead, we assume that the scene is reachable with a manipulator. Hence the problem represents a class of problems of segmentation that occur in an assembly line, bin picking, organizing a desk top and their like. The typical properties of this class of problems are:

1. The objects are rigid. Their size and weight is such that they are manipulable with a suitable end effector. The number of objects in the scene is such that each piece can be examined and manipulated in a reasonable amount of time, i.e. the complexity of the scene is bounded.
2. The scene is accessible to the sensors, i.e. the whole scene is visible, although some parts may be occluded, and reachable by the manipulator.
3. There is a well defined goal which is detectable by the available sensors. Specifically the goal maybe: an empty scene, or an organized/ ordered scene.

The segmentation problem as is specified above is a sub-class of the more general disassembly problem, i.e. taking things apart which may be viewed as a process of getting insight into how to assemble objects, i.e. how to put pieces together. It is not difficult to see that this is how children learn about part/whole relationships and in general about an assembly process. But the question still remains; what perceptual information should be stored when such disassembly process takes place and is it enough for performing the assembly, i.e. the reverse tasks? This problem is what we call the Machine Perceptual Development and is at the heart of this paper.

One may ask how is Machine Perceptual Development related to machine learning? Relevant work on machine learning can be divided into two categories. One involves the application of the neural network paradigm, the other is studies of learning in the AI tradition. The neural net paradigm addresses problems at the low-level perception, learning patterns from the signal, but this approach does not answer the questions of data reduction from a signal that we are proposing. Moreover, we are trying to determine a useful division between "innate" structure and learned properties, that is to say, between a priori and data driven information. The traditional AI approach to learning has most frequently relied too much on



a priori information and has neglected the data driven part. We believe that this approach is too limiting.

## 2 The Two Part Disassembly Problem

We begin with the problem of the two part disassembly. The overall flow diagram of our methodology is as follows: Calibration/Exploration, Disassembly, Assembly. The fundamental issue is the REPRESENTATION. The case still has to be made for new representations that develop during an activity and that respect both the sensory apparatus and the task. **Traditionally, the Computer Vision community** has experimented with geometric CAD models for analysis, arguing that if CAD models are useful for making objects, then they should be equally useful for recognizing them. But such an argument is questionable. A designer creates a CAD model by specifying surface representations with detailed boundaries and explicit dimensions. To represent the internal dimensions, s/he shows cross sections. Finally, s/he specifies both the material and finish of the surface. Thus CAD models reflect how to synthesize an object during both its design phase and its manufacture.

The question is whether this same representation is useful for robotic analysis, i.e., object recognition necessary for disassembly and assembly. We believe the answer is no. First, the limits of sensors determine the limits to which a robotic system can differentiate between different materials, different colors, etc. A robot may not even have the sensors necessary to measure some of the properties that the designer has specified. For example, to distinguish metallic and non-metallic materials, a sensor is needed to measure conductivity. Secondly, the spatial resolution of a sensor limits how well a robotic system can measure spatial details: there is no point in representing a dimension of curvature with tight tolerances if a sensor cannot discriminate it. Thirdly, the noise of the perceptual system determines the minimal discriminability between different categories of objects. Finally, the robot may not know the substance/material of the object it is sensing. Hence it must have an apparatus to find such things out.

What follows in the subsequent sections is: First, the description of the Calibration process which will determine the physical and some geometric characteristics of the material (hardness, coefficient of friction, surface texture, conductivity, spectral properties such as reflectivity, weight/ density and their like). Second, the description of disassembly process and the division of build in procedures versus data driven part. Finally, the test of memory via assembly process.

### 2.1 Calibration/Exploratory Procedures

Unlike much of the current robotics effort we do not assume a priori knowledge of the physical nor geometric properties of objects that we deal with. In order to find out one must have build in capabilities, called Exploratory Procedures (EPs) [9] that seek out different

physical attributes. For this work we shall consider the following EPs: EP that determines the surface reflectance, discriminates between lambertian and highly reflective surfaces [3], EP for determining the hardness of the material and surface texture [12]. Notice that these EPs are static tests, i.e. the object is not manipulated. These EPs will give us the expected range of values for hardness, surface reflectance and surface texture. In the future we will add more attributes, such as electrical and thermal conductivity, measure of elasticity and deformability [11]. Furthermore, weight and density of the material as well moving parts, like objects on hinges, will be explored in a dynamic mode.

## 2.2 The Disassembly/Assembly System

First we shall describe the hardware configuration also shown in Figure 1. For the disassembly/assembly task, the robot is a six degree freedom PUMA 560 manipulator equipped with a range finder and/or a pair of CCD cameras, called the LOOKER and another six degree freedom PUMA 560 manipulator and a hand, called the FEELER. The LOOKER, depending on the need, can also have a color camera system or other non-contact electromagnetic wave measuring detector (infrared is one possibility). The FEELER has a force/torque sensor in its wrist and hand. The hand has three fingers and a rigid palm. Each finger has one and a half degrees of freedom. The sensors on the hand are: Position encoders, force sensors at each joint of the finger, tactile array at each of the finger tip and on the palm, Thermal conductivity sensor on the palm, ultrasound sensor on the outside of the hand. In addition, the hand has access to various tools that it can pick up under its control. Both of the FEELER and the LOOKER are under software control of strategies for data acquisition and manipulation. What are the Logical Components of the System? They are:

1. **SENSOR MODELS** that describe: The range of admissible values, the noise which determines the resolution, the geometry which determines the accessibility of the sensor to the investigated object or of its part.
2. **TASK MODEL**: In this case: a two part decomposition/separation.
3. **PARAMETERS**: About the physics/geometry of an object obtained through calibration EPs.
4. **MANIPULATION PROCEDURES**: such as: Push, Pull, Lift, Press, Turn, Twist, Grasp, Squeeze.
5. **GEOMETRIC PROCEDURES**: Shape description, especially detection of discontinuities, where is the binding force, size (length, area, volume) determination.
6. **CONTROL STRUCTURE**: (State, Actions), Priorities if more than one possible action, (here one may consider some cost/benefit function to make the right choice). Priority of sensing: how to start? (here we start with vision!). Detection of the goal state, i.e. two separate parts.

The Block diagram reflecting the logical components for disassembly/assembly is shown in Figure 2. This diagram is very similar to the one used in Tsikos's Ph.D thesis [13] for segmenting a complex scene. We have shown that:

1. Segmentation of an arbitrary scene requires not only a visual sensor, but also some manipulation actions, such as pushing, pulling, grasping and their like.
2. The interaction between the sensors and manipulation and the scene can generally be sufficiently modeled by a finite state, non-deterministic Turing Machine.
3. The critical consideration is the testability of the goal state. (In Tsikos' case it was an empty scene.)

## 2.3 The Disassembly Process

As a test for our system, consider a peg-and-hole problem shown in Figure 3. It is a test bed with the same shapes of the top of the peg but with differing holes (square, circular, or none) Figure 3d, 3c, 3a and with varying surface finish of the peg (smooth as shown in Fig. 3c and 3d, and threaded as shown in Figure 3b). This fixture has been designed so that we can test several combinations of manipulative actions. The general priority schema of control is as follows:

1. LOOK. Remember: Position and shape. Start with vision, identify the surface discontinuity of the peg-head *vis-a-vis* the hole surface, find the position, orientation, surface normal, and shape of the peg-head.
2. GRASP. Remember: Position and grasping force. After vision follow up with grasping in preparation for manipulation. The grasping procedure includes the limitations of the end-effector, i.e. this procedure utilizes the parameters obtained through calibration EPs and from the previous step which provides information on geometry of the peg-head.

Our initial experiments were carried out using a parallel jaws gripper instrumented with force/torque sensors and tactile arrays. The goal of the grasp action is to verify correct grasp of an orthogonal parallelepiped peg-head. We define correct grasp to be a two PLANE contact between the jaws and the peg-head such that the forces and torques exerted on the sensors are of approximately equal magnitude and opposite sign.

We use a binary search procedure in three space to verify and/or correct the position, orientation, and surface normal as computed by vision. The first step in this procedure is to make an initial grasp of the peg-head. In the general case the initial grasp will be two POINT contacts between the gripper jaws and the surfaces of the peg-head, See Figure 4a. Then we measure forces and torques. Using the sign and magnitude of these measurements we un-grasp the object, reorient the gripper and attempt another grasp

until we have (in general) a two LINE contact between the jaws and the peg-head, as illustrated in Figure 4b. At each iteration, the changes in gripper orientation are one half of the previous step. This procedure continues until we have a two PLANE contact, see Figure 4c, and the forces/torques are of equal magnitude and opposite sign.

3. MANIPULATE-PULL. Remember: Direction and magnitude of pulling force while in the hole, and the positions during the departing motion (change in the magnitude of the pulling force). This procedure adaptively (using force feedback) pulls the peg by finding the direction which minimizes the reactive force. This procedure uses differential force feedback to subtract the grasping forces, recorded during the grasping phase.
4. OBSERVE the action using vision during manipulation. Remember: Shape, size and position of the two separating parts. An alternative to using vision during manipulation is to use a move until free primitive action that moves the manipulator slightly in a direction normal to the pulling force. If the disassembly of the peg is not yet complete then forces/torques will be exerted on the sensors. The system then returns back to the previous state, and continues with the manipulate- pull action.
5. GOAL STATE CHECK. If the two parts are separated then the goal state has been reached and stop. Notice that there are two ways to measure that the goal state has been reached. One is to use information from the contact sensors i.e. move until free, and the other is to use vision during manipulation to detect separation. In this work we use the former and we plan to integrate the latter soon. Notice that both methods allow us to measure the unknown length of the peg. Only vision, however, can measure the shape of the peg as well as the shape of the hole after the disassembly is complete. This is important in the general disassembly problem.

## 2.4 The Assembly Process

The fundamental question in disassembly is: Did the system remembered enough? Consider reversing the above described process: The FEELER is holding the head of the peg and we have stored the position and shape of the hole. Hence unless something has changed the FEELER can approach the hole without the LOOKER. The insertion process is the reverse of manipulate-pull. The goal state is determined by the length of the peg, that was remembered by the LOOKER after separation of the two parts. We conclude that at least in this test case the system remembered enough to pass the test.

## 3 Conclusion

We have defined and outlined our long-term thinking and investigations on Machine Perception that leads us to the latest research program of understanding (Machine Perceptual

Development). This is an outgrowth of our research on Active Perception, which views perceptual activity as an active process of SEEKING INFORMATION. Naturally this is not just blind pickup of any information. The system must protect itself by imposing some economy rules [8]. Even if the perceptual system receives overabundant amounts of information, again for economy reasons it must be selective in what it stores. Hence the fundamental problem remains: The REPRESENTATION issue. What is it that the system must have to seek, measure, and select in order to be able to move and manipulate?

Somewhat similar ideas appear in the work of Donald [5-7], and Pertin-Trocac and Puget [10]. They consider a manipulation program automatically generated by a planner according to spatial and geometric criteria and ignoring uncertainties. Such a program is correct only if, at each step, uncertainties are smaller than the tolerance imposed by the assembly task. They propose an approach which consists in verifying the correctness of the program with respect to uncertainties in position and possibly modifying it by adding operations in order to reduce uncertainties. These two steps based on a forward and a backward propagation borrowed from formal program proving techniques are described in a general framework suitable for robotic environments. Forward propagation consists in computing successive states of the robot world from the initial state and in checking for the satisfaction of constraints. If a constraint is not satisfied, backward propagation infers new constraints on previous states. These new constraints are used for patching the program.

However, we differ in more than one ways from their approach. The most important difference is the ultimate goal, that is we are interested in the perceptual data reduction mechanisms rather than in a general plan of a process. We have posed these questions in the framework of disassembly of one object into two parts and tested the selected, remembered representation by reversing the process, i.e assembly. Our results are only very modest but we believe that they are encouraging!

## 4 Acknowledgements

This work was supported in part by: The U. S. Air Force grant AFOSR F49620- 85-K-0018, U. S. Army grant DAAG-29-84-K-0061, NSF grant CER/DCR82-19196 Ao2, NASA grant NAG5-1045, ONR grant SB-35923-0, NIH grant NS-10939-11 as part of the Cerebro Vascular Research Center, NIH grant 1-RO1-NS-23636-01, NSF grant INT85-14199, NSF grant DMC85-17315, ARPA grant N0014-88-K-0632, NATO grant 0224/85, by DEC Corp., IBM Corp. and LORD Corp.

## 5 References

1. Bajscy, R. K. (1982) What Can we learn from One-Finger Experiments. U.S. - France Seminar in Robotics, Paris, May, 1982.

2. Bajcsy, R. K. (1988) Active Perception. Proceedings of the IEEE on Computer Vision, August, 1988.
3. Bajcsy, R. Wohn, K. and Lee, S. W. (1988) Exploratory Procedures For Computer Vision. Submitted to the IEEE Transactions on Systems, Man and Cybernetics. October, 1988.
4. Bajcsy, R. K. and Tsikos C. J. (1987) Perception Via Manipulation. R. Bolles (Editor), 4th ISRR, Santa Cruz, California, August 9-14, 1987.
5. Donald, B. R. (1987). Towards Task-Level Robot Programming. Technical Report 87-878, Computer Science Dept., Cornell University, 1987.
6. Donald, B. R. (1988) The Complexity of Planar Compliant Motion Planning under Uncertainty. Proc. ACM Symposium on Computational Geometry, Urban, Ill., 1988.
7. Donald, B. R. (1988). Planning Multi-Step Error Detection and Recovery Strategies. Proc. IEEE International Conference on Robotics and Automation, Philadelphia, PA., April, 1988.
8. Hager G. (1988) Active Reduction of Uncertainty in Multi-Sensor Systems. Ph.D. Dissertation, Computer and Information Science Department, University of Pennsylvania, July, 1988.
9. Klatzky R. L., and Lederman S. There is more to touch than meets the eye: The Salience of Object Attributes for Haptics with and without Vision. Journal of Experimental Psychology, Vol. 116, No. 4, 1987. pp. 359-369.
10. Pertin-Troccaz, J. and Puget, P. (1987). Dealing with Uncertainty in Robot Planning using program proving techniques. R. Bolles (Editor), 4th ISRR, Santa Cruz, CA, August 9-14, 1987.
11. Sinha P. (1989) Haptic Exploration for Robots. Personal Communications, January, 1989.
12. Stansfield, S. (1987). Visually-Guided Haptic Object Recognition. Ph.D. Dissertation, Computer and Information Science Department, University of Pennsylvania, October, 1987.
13. Tsikos, C. J. (1987). Segmentation of 3D Scenes Using Multi-Modal Interaction Between Machine Vision and Programmable Mechanical Scene Manipulation. Ph.D. Dissertation, Computer and Information Science Department, University of Pennsylvania, December, 1987.
14. Tsikos, C. J. and Bajcsy, R. K. (1988). Segmentation Via Manipulation. Submitted to the IEEE Journal of Robotics and Automation, June, 1988.

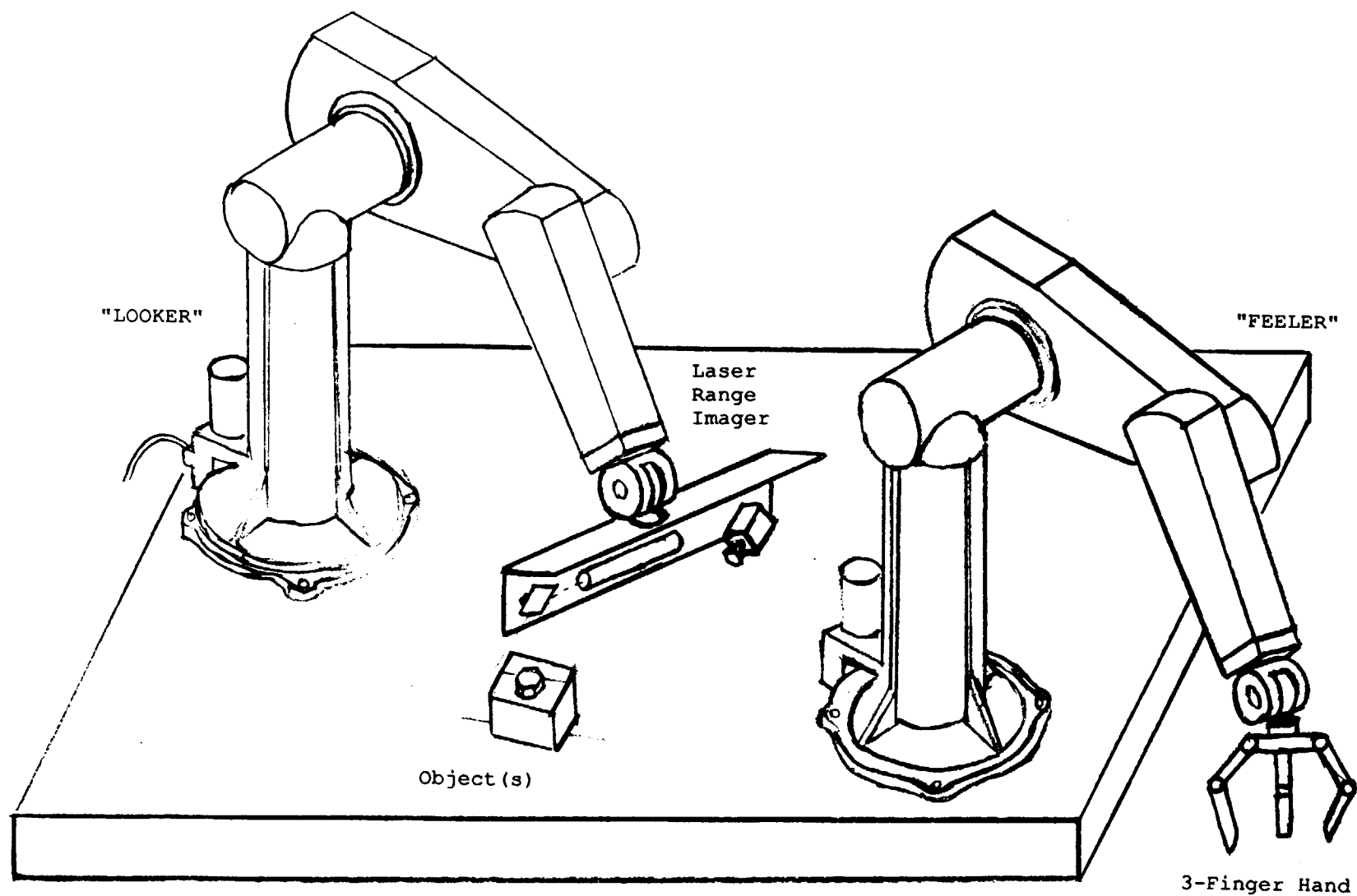


Figure 1. The Disassembly/Assembly System Hardware.

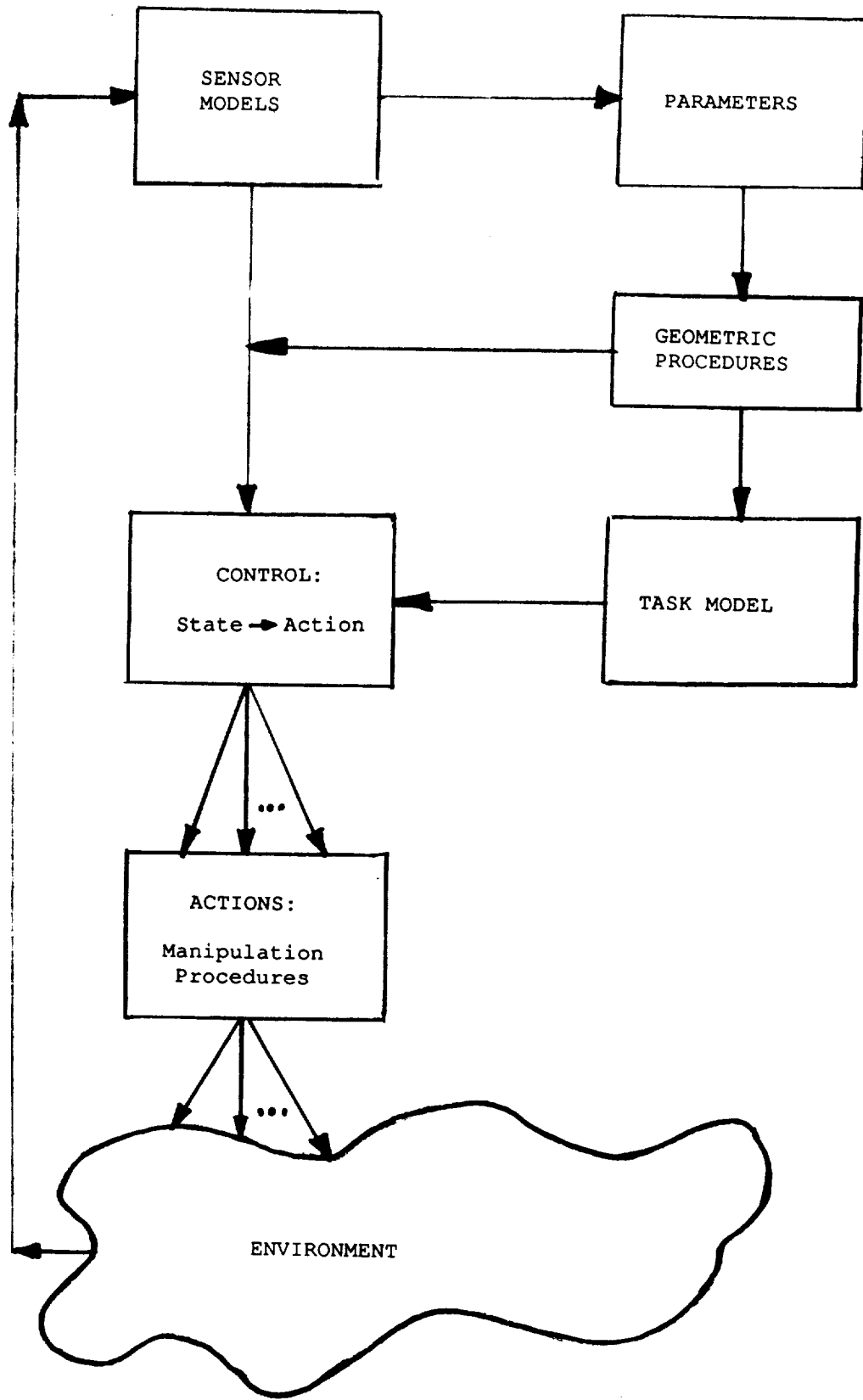


Figure 2. The Disassembly/Assembly System Block Diagram.



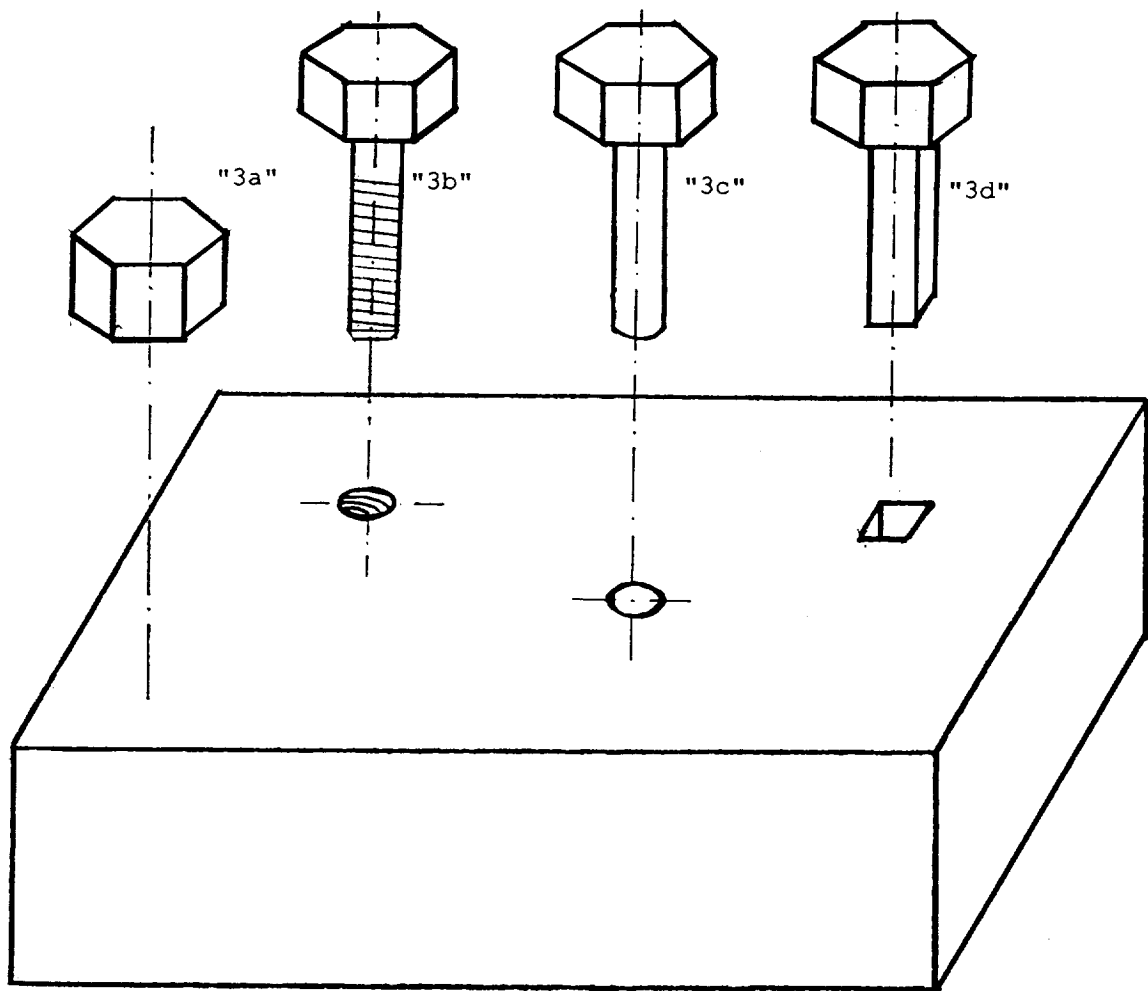


Figure 3. Various Instances of the Two-Part Disassembly/Assembly Problem.

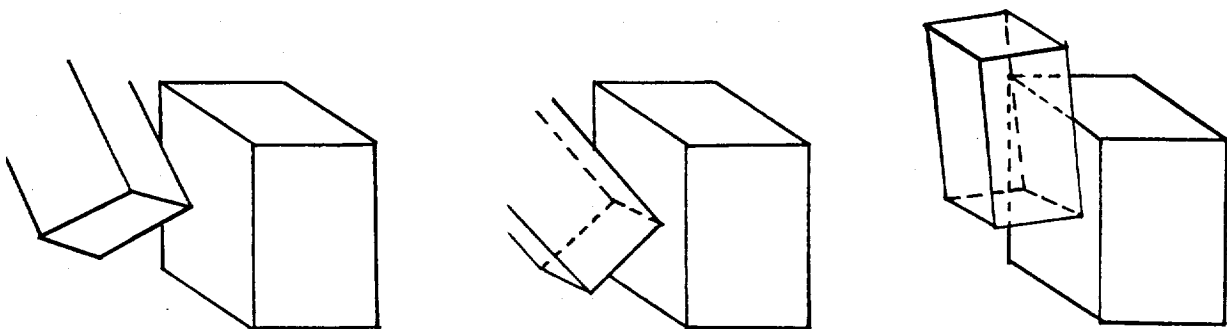


Fig. 4a. POINT Contact. Fig. 4b. LINE Contact. Fig. 4c. PLANE Contact.

Figure 4. Grasp with POINT, LINE, and PLANE Contacts. (Only one finger shown).



# TECHNIQUES AND POTENTIAL CAPABILITIES OF MULTI-RESOLUTIONAL INFORMATION (KNOWLEDGE) PROCESSING

A. Meystel

Department of Electrical and Computer Engineering  
Drexel University  
Philadelphia, PA 19104

## Abstract

A concept of nested hierarchical (multi-resolutional, pyramidal) information (knowledge) processing is introduced for a variety of systems including data and/or knowledge bases, vision, control, and manufacturing systems, industrial automated robots, and (self-programmed) autonomous intelligent machines. A set of practical recommendations is presented using a case study of a multiresolutional object representation. It is demonstrated in the paper, that any intelligent module transforms (sometimes, irreversibly) the knowledge it deals with, and this transformation affects the subsequent computation processes, e.g. those of decision and control. Several types of knowledge transformation are reviewed. Definite conditions are analyzed in this paper, satisfaction of which is required for organization and processing of redundant information (knowledge) in the multi-resolutional systems. Providing a definite degree of redundancy is one of these conditions.

**Key Words:** *Abstraction, Generalization, Image Analysis, Interpretation, Knowledge, Multigrid Relaxation, Multiresolutional, Pyramidal, Redundancy, Representation.*

## I. Introduction

A concept of nested hierarchical (*multi-resolutional*, pyramidal) information (*knowledge*) processing (MRKP) is becoming increasingly important in the area of intelligent machines including robotics, computer vision, and knowledge-based material processing. *Multiresolutional Knowledge Representation is defined as the union of all monoresolutional representations.* Monoresolutional representation is understood as a representation of the particular set of reality <sup>1</sup> at a resolution commensurable with the subset of required measurements and activities.

The main idea of this concept is that the applicable model of a system cannot be built unless this system is considered simultaneously at several levels of resolution <sup>2</sup>. Resolution is defined as a minimum volume of the state space that is distinguishable within a particular system of representation. This minimum volume is called *tessella* (in Latin - minimal element of a mosaic), and organization (discretization, quantization) of the state space is called *tesselatrin* if a particular size of tessella is being used efficiently as an element for building all descriptions of interest.

---

<sup>1</sup> Set of interest

<sup>2</sup> Thus, MKR is not equivalent to the "syntactic" representation known in the theory of pattern recognition. The latter does not require that each level of the syntactic graph to be necessarily a complete representation of the set of interest. However, the methods of dealing with these representations are similar. This problem should be discussed separately.

Many characteristics (properties, variables) make sense only at a particular level of resolution, and do not need to be reflected at a higher or lower resolutions. In the meantime most of the existing control problems cannot be solved only within one resolution level. Thus, a concurrent consideration of the system at several resolution levels is required, and the redundant representation is justified in which the "same" thing is represented several times with different resolution. Utilization of MRKP is discussed in [1], and a brief survey of literature on multiresolutional models of knowledge organization is given in Section 2.

A notion of **multiresolutional knowledge representation (MKR)** is introduced for a variety of systems including data and/or knowledge bases, vision, control, and manufacturing systems, industrial automated robots, and (self-programmed) autonomous intelligent machines. Most of these applications are actually, or presumably utilizing **intelligent modules** with decision making capabilities, (or human operators performing similar functions). The structure of intelligent module is described in [1], (this is a system which exercises intelligent control similar to what in AI literature is called sometimes an *intelligent agent*). MKR is derived directly from the entity-relational representation of a system. This representation is using the following postulates of representation:

*Postulate 1.* Any representation is derived from a verbal<sup>3</sup> description<sup>4</sup>.

*Postulate 2.* Any verbal description transmits information about labels (words) that can be interpreted within some global thesaurus.

*Postulate 3.* Relations among the labels can be determined from the same description or from the set of related descriptions (context<sup>5</sup>).

*Postulate 4.* Relations among the labels can be numerically evaluated in the scale generated by a metric meaningful for considering a particular label.

*Postulate 5.* The set of interest at a particular resolution level has a multiplicity of corresponding sets at all other levels of higher as well as lower resolution; each of these sets represents a concrete set of reality with resolution pertained to the set; only all of them together adequately (completely) represent of the concrete set of reality.

The first three postulates are establishing a graph representation for the system of interest. This graph includes all levels of resolution since it contains not only the systems represented but also their components, and components of their components, and so on. The last postulate presumes that the classes can be recognized among the multiplicity of labels, of those commensurable labels,

<sup>3</sup> The word "verbal" is used in a very general sense. Of course, it means "expressed in words". Obviously, it includes any process of discretization when the signal is assigned a discrete number for further utilization in the algorithm where the number is used as a value for the signal (so, the signal, or in general, a variable is considered to be a *word with a value*). However, it includes also any process with no discretization since in the analogous systems we can use a loop in which a variable (a word with a value) is operating with no discretization required.

<sup>4</sup> I.e. even we have a pictorial description either we transform it into words before using it, or it by itself is a result of transforming the verbal description into a pictorial illustration. Also, the word description is related not to a particular description which almost definitely is always incomplete, but rather to a **representative set of verbal descriptions** which is considered to be representative by the experts in this area. This set can include scientific papers, articles from trade magazines, technical reports of industrial companies, and/or universities, as well as interviews with the experts.

<sup>5</sup> Context is presumed even if the verbal descriptions are only implicit ones and exist as a potential set of descriptions within the experience of experts. (Another problem is that the descriptions generated by these experts are not necessarily conducive to the transfer of objective information about their experience).

i.e. belonging to the same space of consideration. The subset of commensurable words we will call a **scope**. Figure 1 illustrates the entity-relational graph ("a"), the ability to view this graph in a variety of scopes (e.g. I-"function", II-"perception", III-control<sup>6</sup>), and the ability to redraw it in such a way as to reflect this classification into this set of scopes ("b").

One can see that the structure can be visualized as a set of the interrelated scope graphs  $\bigcup G_i$ ,  $\bigcup G_i R G_j$ ;  $i, j = I, II, III$ ,  $i \neq j$ , where  $R_{ij}$  is a relation among the elements of the graphs. Each of the scope graphs has a set of vertical (hierarchical) connections of the resolution levels and this set of connections is called a *hierarchy of the scope*. Within each level of resolution an entity-relational graph (*tessellatum*<sup>7</sup>) exists which represents all entities and relations among them at a particular resolution (or accuracy which is characterized by a minimum cell of distinguishability (*tessella*). There are no hierarchies within a tessellatum: all entities that can be partitioned are partitioned and their parts belong as entities to a lower level tessellatum. All tessellata belong to a particular hierarchy and are being considered together with it:

$$G_i = \bigcup_{ik} T_i^k R T_i^{k+1}, \quad k=1, \dots, n, \quad i=I, II, III \quad (k \text{ is a number of resolution levels}).$$

Each of the is unifying the set of inclusions for the tessellata

$$G_i = \bigcup_g (T_i^k \supset T_i^{k+1} \supset T_i^{k+1} \supset \dots \supset T_i^{k+1})$$

where the inclusions are meant to represent the relations **R**. These relations are of a special meaning: they reflect the fact that the entities and relations of the lower resolution levels can be obtained from the corresponding entities and relations of the higher resolution level via mechanism of generalization (or abstraction). Or, in other words: any tessellatum of the higher resolution level can be transformed into the tessellatum of the lower level via mechanism of *generalization (abstraction)*. This is why these inclusions have an index "g": it reflects that a special set of rules is presumed which provides this inclusion generating transformation of generalization (abstraction).

A set of all hierarchies with all tessellata related to each of the hierarchies forms a *heterostructure* (see **D-structure** in [2]).

A number of laws of multi-resolutional information (knowledge) organization and processing, enable us to deal with the subsystem of information (knowledge) independently from the associated subsystem of decision making the latter must be taken in account at the stage of designing the algorithms of information (knowledge) processing. In this paper we will focus only on the general matters which are important for the whole variety of methods of Multiresolutional Knowledge Processing (MRKP). This variety is surveyed in Section 2. Section 3 analyses a Case of MKR. Techniques of MRKP are discussed in Section 4. Finally the potential capabilities of MKR for MRKP are described in Section 5.

## 2. Overview of the Situation in the Area of MRKP

MKR and associated techniques of MRKP was rapidly developing during past two decades from three different views: hardware MKR, visual images MKR, and algorithms MKR (with fuzzy boundaries). Firstly, it has been realized that using effectively multilevel, multilanguage structure of

<sup>6</sup> These three types of scope are typical for making theories about many objects of external world because they actually exhaust the areas of interest and application.

<sup>7</sup> tessellatum- a mosaic floor composed of a multiplicity of minimal elements or tessella (from Latin).

a computer is possible only if this multilevel structure is explicitly, consciously associated with the multilevel (multiresolutional) organization of the World constructed by methods of aggregation (generalization, abstraction) and decomposition (instantiation). This became clear in CAD/CAM area, and a number of multilevel (multiresolutional) hardware descriptions appeared as well as methods of reasoning about World [3,4]. This area is linked with the problem of partitioning multiprocessor systems in order to achieve maximum of efficiency. Proper distribution of resolution among subsystems should provide the best utilization of equipment [1,5].

Another MKR problem adjacent to the problem of hardware partitioning was the following: how to partition something that has not been previously assembled, (e.g. partitioning of a curve) [6]. It was determined that the following factors must be taken in account: digitization and/or resolution of representation on hand, existence of multiple "views", and the set of attributes utilizable for describing the object to be partitioned. Linkage of all these approaches is undeniable to the "frame approach" from AI, and aggregation/decomposition methodologies of the earlier scientists belonging to the school of thought of General Systems Theory (e.g. see [7]). A method of multiresolutional curve representation is presented in [8] which is a good illustration of the definition of the MKR, and of the *generalization* as a major technique which transforms the representation given at a higher level of resolution into the lower level of resolution creating a hierarchy of generalizations (or abstractions).

Pyramid theories of image processing and interpretation have been promulgated during the last two decades in a multiplicity of well known books and papers by L. Uhr, E. Riseman, A. Hansen, S. Tanimoto, T.Pavlidis, M.Levine, R.Bajcsy, P.Burt, A.Rosenfeld [9-14]. The idea of generalization of information from level to level is presented and developed in all of their papers, and a variety of methods is proposed for solving practical problems under these conditions. Most of them are boiling down to decomposition of entities of the upper level into the set of entities of the lower level in such a way as to have the whole level given at a definite particular resolution consistent with the context determined by the focus of attention at this level as illustrated in this sequence:

**level of resolution  $\Rightarrow$  detail ( tessella)  $\Rightarrow$  focus of attention  $\Rightarrow$  context  $\Rightarrow$  level of resolution**

Interestingly enough, the well known quadtree structure [15] is not a multiresolutional structure in a sense that the accuracy of representation is the same at each level: the highest available accuracy of the level with the highest resolution (the lowest level of consideration). Only recently, there was an attempt to fuzzify the upper levels images when the problem of planning was attempted using quadtree as a MKR system [16]. Truly MKR approach with using all tessellata for planning was successfully employed in [33].

Partitioning driven by a linguistic description leads to MRKs which are instrumental in shape description. It turned out that the set of hierarchical connections (those of  $G_i$  type) forms a "skeleton" that can be used as a good enough "syntactic" <sup>8</sup> representation of various complicated shapes [18,19]. This phenomenon seems to have explanations within the principles of human perceptions reflected in the biological structure of vision system. This view was reflected in the multiresolutional model of the visual receptive fields [20]. Multiresolutional representation turned out to be useful also for image segmentation and to region matching [21, 22].

MRKP is kindred to the fractal methodology of world representation [23]. Multiple-scale based approach to image representation and analysis [24] together with fractal-based techniques is actually

<sup>8</sup> The problem should be addressed separately of reconciling the multiresolutional approach with its MRK systems with the well known syntactic methods of pattern description and pattern recognition (for example, like in [17]). The conventional syntactic representation and the representation (like in [17]) and MRK should not be confused.

application of the set of ideas characteristic for MKR. Here we are dealing with *simultaneous representation of all images at all resolutions* when the mechanism of generalization (or abstraction) is imposed upon the system by an external mathematical model.

Finally, the last group of MRKP results is related to the multiresolutional algorithms. Somewhat interlaced with the fractal methodology are the algorithms of continued fractions [25,26]. Multiresolutional relaxation algorithms have been recommended for efficient dealing with texture [27]. A consistent and complete overview of the multigrid relaxation algorithms for image processing can be found in [28].

### 3. A Case Study: Multiresolutional Representation of a Chair

Chair is a tempting example for illustrating the techniques of MRKP. Many researchers were choosing this object even in the area of multiresolutional information processing (F. Mokhtarian, R. Bajcsy, et al). In Figures 1-10 some of the illustrative material is presented. More detailed description of operations is given in [32].

### 4. Discussion of the Techniques of MRKP

As one can see from Figure 1, and the illustrating case, the core of operation of the intelligent module does not differ from the process of the **automated theory generation (ATG)**. The latter was first tackled in [29] and then was furtherly developed in [30] and other works. It is important to emphasize that any **process of representation is based upon theory generation**. Like in ATG, the subsystem of representation is supposed to synthesize a consistent system of tessellata constructed at different resolutions and transformable one into another. This synthesis can be performed in a different way depending on initial problem specifications. We give two examples a) for the case of "well known systems"<sup>9</sup> (i.e. knowledge is available if needed and all possible interpretations can be found), and b) for the case of a system with high resolution information available<sup>10</sup>.

#### *Case 1. "Well Known Systems"*

- Step 1: present the description of a system including its function, its component, and its operation,
- Step 2: explain the meaning of the components, and the relations among them (ER graph),
- Step 3: perform steps 1 and 2 for the components of the system <sup>11</sup>, and continue this down to the *meaningful* high resolution level,
- Step 4: determine (discover?) generalizations within the results of Steps 1 through 3 activities, which can simplify understanding, memorization, utilization, computation, and so on, of the system and its components. These generalizations can be in the form of rule tables, mathematical formulas, geometrical analogies, computational algorithms, and any another form applicable in the domain of interest.

#### *Case 2. Systems With High Resolution Information Available*

- Step 1: present the expected description of a system including its expected function, its expected component, and its assumed operation,
- Step 2: prepare the possible structure of interpretation for the components, and the relations among them (ER graph) at all meaningful resolution levels,
- Step 3: perform steps 1 and 2 for the components of the system within the expected tessellata, and continue this down to the *given* high resolution level,
- Step 4: determine (discover?) generalizations applicable within the results of Steps 1 through 3

---

<sup>9</sup> This case can be identified with those known in the CAD/CAM, FMS, etc.

<sup>10</sup> Applicable in computer vision systems.

<sup>11</sup> The system is presumed to allow consecutive decomposition ("consecutively decomposable system").

activities<sup>12</sup>.

- Step 5. Apply the generalization required to each tessellatum bottom up and verify consistency of the representation.

Even in the case of image processing, generalizations are not to be sought in a form of some simple algorithm uniformly applied to each tessellatum of the system (e.g. as a low pass filter recommended in a number of papers on multiresolutional representation of images, or as an algorithm of quadrics-type universal tessella of the level).

Definite conditions should be satisfied for organization and processing of redundant information (knowledge) in the multi-resolutional systems. Providing a definite degree of redundancy is one of these conditions. A definite set of rules of incorporating the redundant information (knowledge) must be applied for the system proper functioning. The significance of proper dealing with redundancy of information (knowledge) is often overlooked. Several operators are discussed in [1] implicitly using redundancy of information (knowledge): generalization (abstraction), focusing of attention, etc. The following relationship is important for computer simulation of perceptual processes: among the total volume of information (knowledge)  $I_{TC}$  (for totality associated with the problem of control), and the size of minimal cell of distinguishability  $\Delta$  required by the customer specifications. On the other hand, the number of resolution levels in the nested hierarchical system depends on the ratio  $I_{TC}/\Delta$ . Phenomena of multi-resolutional redundant perceptual organization are linked with the phenomena of error propagation (see [31]).

## 5. Potential Capabilities and Perspectives of MRKP

Any intelligent module transforms (sometimes, irreversibly) the knowledge it deals with, and this transformation affects the subsequent computation processes, e.g. those of decision and control. Several types of knowledge transformation are reviewed. One of them called knowledge filtering (KF) can be characterized by its volume and rate. The detrimental effect of KF can be compensated by the corresponding level of knowledge redundancy (and by the subsequent redundancy of decision making processes, followed by the action redundancies as well).

MKR allows for coding the system as a whole and not as a result of selecting only its limited subset. This allows for a harmonious control of a system. In [34] an example is described of using MRKP system for intelligent control of the OSPREY process in the metallurgy. Another system is now in the process of development for a plasma deposition machine.

## References

1. A. Meystel, "Theoretical Foundations of Planning and Navigation for Autonomous Robots", *International Journal of Intelligent Systems*, Vol. II, No. 2, Summer 1987, p.p. 73-128
2. A. Meystel, "Intelligent Control in Robotics", *Journal of Robotic Systems*, 1988
3. B. Moszkowski, "A Temporal Logic for Multilevel Reasoning about Hardware", *Computer Magazine*, February, 1985, p.p. 10-19
4. S. Y. H. Su, C. Huang, P. Y. K. Fu, "A New Multilevel Hardware Design Language and Translator", Proc. of 5th Conference on Computer Hardware Description Languages, Kaiserslautern, West Germany, 1981, p.p. 155-169

<sup>12</sup> As in the Case 1 these generalizations are expected to simplify understanding, memorization, utilization, computation, and so on, of the tessellatum and its components. They can be in the form of rule tables, mathematical formulas, geometrical analogies, computational algorithms, and any another form applicable in the domain of interest.



5. R. Agrawal, H. Jagadish, "Partitioning Techniques for Large-Grained Parallelism", *IEEE Transactions on Computers*, Vol. 37, No.12, 1988, p.p. 1627-1634
6. M. Fischler, R. Bolles, "Perceptual Organization and Curve Partitioning", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-8, No.1, 1986, p.p. 100-106
7. M. Mesarovic, "Foundations for a General Systems Theory", Ed. by M. Mesarovic, Views on General Systems Theory, Proc. of the 2-nd Systems Symposium at Case Institute of Technology, Wiley, NY 1964
8. W. Kropatsch, "Curve Representation in Multiple Resolutions", Proc. of the 8-th Int'l Conference on Pattern Recognition, Paris, France, October 27-31, 1986, p.p. 1283-1285
9. L. Uhr, "Layered "Recognition Cone" Network That Preprocess, Classify, and Describe", *IEEE Transactions on Computers*, Vol.C-21, 1972, p.p. 758-768
10. A.R. Hansen and E. M. Riseman, Preprocessing Cones: A Computational Structure for Scene Analysis, Technical Report of the Computer and Information Science Department, University of Massachusetts, 1974
11. S. Tanimoto, T. Pavlidis, "A Hierarchical Data Structure for Picture Processing", *Computer Graphics Image Processing*, No.4, 1975, p.p. 104-119
12. M. D. Levine, J. Leemet, "A Method For Nonpurposive Picture Segmentation", Proc. of the 3-rd Int'l Joint Conference on Pattern Recognition, Coronado, CA 1976, p.p. 494-498
13. D. Rosenthal, R. Bajcsy, "Visual and Conceptual Hierarchy - A Paradigm for Studies of Automated Generation of Recognition Strategies", Ed. A. Rosenfeld, Multiresolutional Image Processing and Analysis, Springer Verlag, Berlin, 1984, p.p.60-76
14. P. J. Burt, "The Pyramid as a Structure for Efficient Computation", Ed. A. Rosenfeld, Multiresolutional Image Processing and Analysis, Springer-Verlag, Berlin, 1984, p.p. 6-36
15. H. Samet, "The Quadtree and Related Data Structures", *ACM Comput. Surv.*, Vol.16, No.2, June 1984
16. S. Kambhampati, L. Davis, "Multiresolutional Path Planning for Mobile Robots", *IEEE Journal of Robotics and Automation*, Vol. RA-2, No. 3, 1986
17. K.-S. Fu, "Syntactic Pattern Recognition", Eds. T. Y. Young, K.-S. Fu, Handbook of Pattern Recognition and Image Processing, Academic Press, Orlando Fl, 1986, p.p. 85-118
18. S. M. Pizer, W. R. Oliver, "Hierarchical Shape Description Via the Multiresolutional Symmetric axis Transform", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-9, No. 4, 1987, p.p. 505-511
19. A. Dill, M. Levine, P. Noble, "Multiple Resolution Skeletons", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-9, No. 4, 1987, p.p.495-504
20. J.-P. Crettez, S. Tanimoto, "Perceptual Constancy and the Multi-Layer Visual Model: Position and Size Invariance", Proceedings of the Workshop on Computer Vision, Washington DC, 1984, p.p. 518-52.
21. W. Grosky, R. Jain, "A Pyramid Based Approach to Segmentation Applied to Region Matching", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-8, No. 5, 1986, p.p. 639-650
22. R. Ohlander, K. Price, D. Raj Reddy, "Picture Segmentation Using a Recursive Region Splitting Method", *Computer Graphics and Image Processing*, Vol. 8, 1978, p.p. 313-333

23. B. Mandelbrot, The Fractal Geometry of Nature, Freeman and Co, NY 1983
24. F. Mokhtarian, A. Mackworth, "Scale-Based Descriptions and Recognition of Planar Curves and Two-Dimensional Shapes", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-8, No. 1, 1986, p.p. 34-43
25. U. Cheng, "On the Continued Fraction and Berlekamp's Algorithm", *IEEE Transactions on Information Theory*, Vol. IT-30, No.3, 1984, p.p. 541-544
26. N. Blachman, "The Continued Fraction as an Information Source", *IEEE Transactions on Information Theory*, Vol. IT-30, No.4, 1984, p.p.671-681
27. F. Cohen, D. Cooper, J. Silverman, E. Hinkle, "Simple Parallel and Relaxation Algorithms for Segmenting Textured Images Based on Noncausal Markovian Random Field Models", Proc. of the 7-th Int'l Conference on Pattern Recognition, Montreal, Canada, 1984, p.p. 1104-1107
28. D. Terzopoulos, "Image Analysis Using Multigrid Relaxation Method", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-8, No. 2, 1986, p.p. 129-139
29. S. Amarel, In Principles of Self-Organization, Trans. of the University of Illinois Symposium on Self-Organization, June 8-9, 1961, Eds. H.Von Foerster, G. Zopf, Jr., Pergamon Press, Oxford, 1962
30. R. Davis, D. Lenat, Knowledge-Based Systems in Artificial Intelligence, McGraw Hill, NY 1982
31. A. Meystel, "Multiresolutional Models of Uncertainty Generation and Reduction", (in this Proceedings).
32. A. Meystel, G. Chen, Multiresolutional Structure of Assembled Objects, Technical Report, Drexel University, Philadelphia, PA 1989
33. P. Graglia, Path Planning in the Multiresolutional Traversability Space, MS Thesis, Drexel University, 1988
34. D. Apelian, A. Meystel, "Knowledge Based Control of Material Processing" (to be published in Proceedings of the 118-th TMS Annual Meeting, February 27-March 2, 1989, Las Vegas, Nevada; Volume on Symposium "Metallurgical Processes for the Year 2000 and Beyond", Publ. by The Minerals, Metals, and Materials Society, 1989).

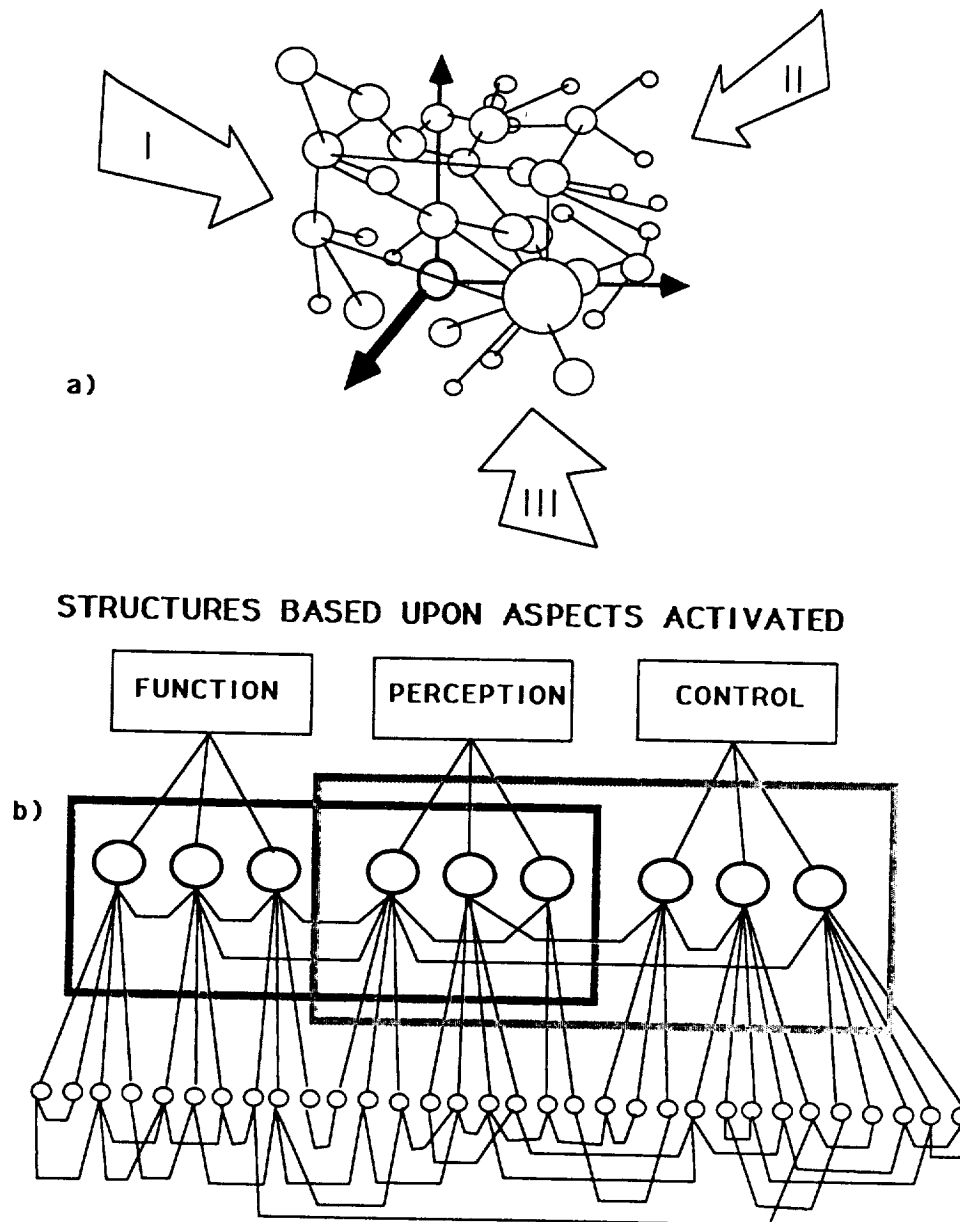


Figure 1. Full Heterostructure of the Chair

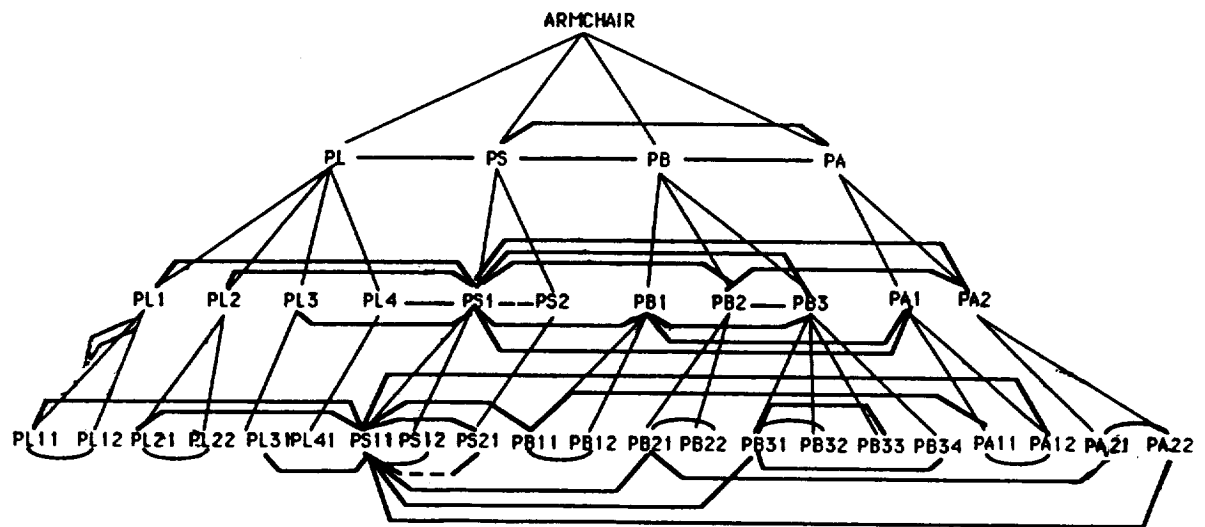


Figure 2. Graph of the Scope of perception

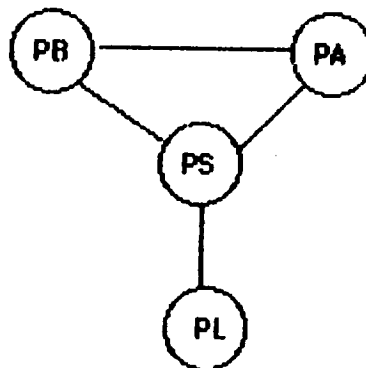


Figure 3. Tessellatum of the LR- level (low resolution) of the Perception Scope

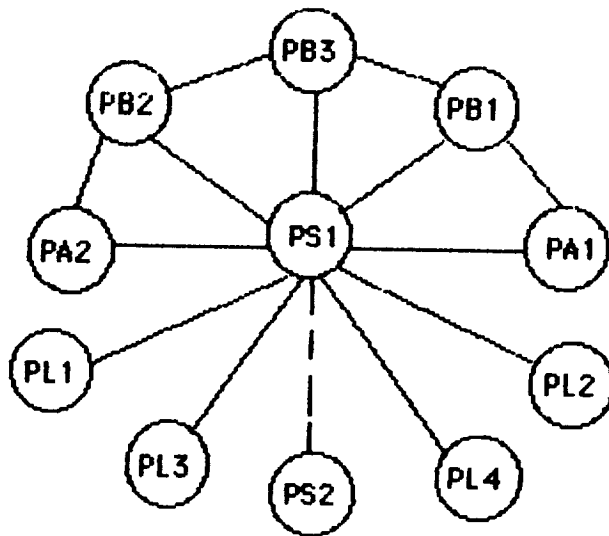


Figure 4. Tessellatum of the HR- level (high resolution) of the Perception Scope

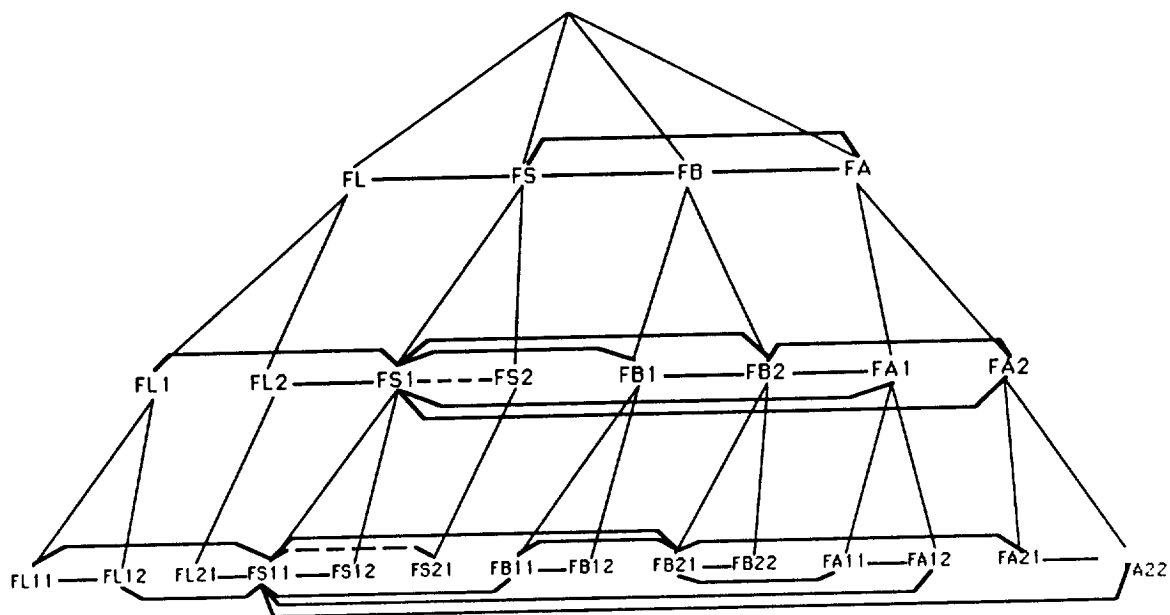


Figure 5. Graph of the Scope of Function

Figure 6. Tessellatum of the LR of the Scope of Function

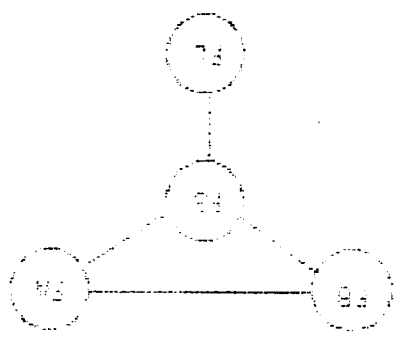
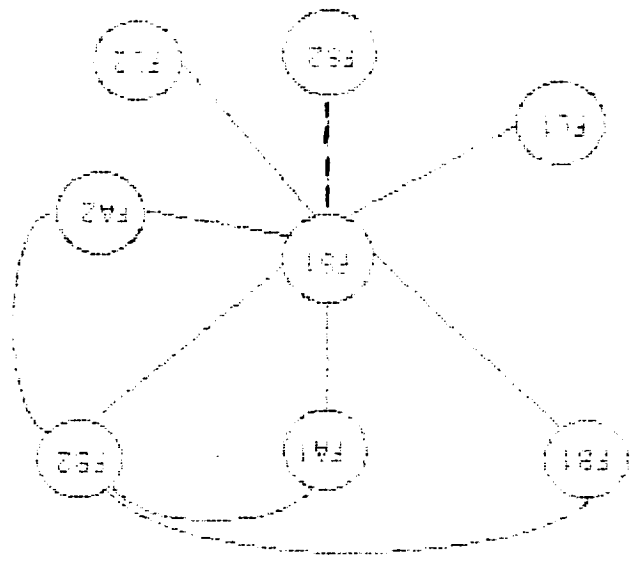


Figure 7. Tessellatum of the HR level of the Function Scope



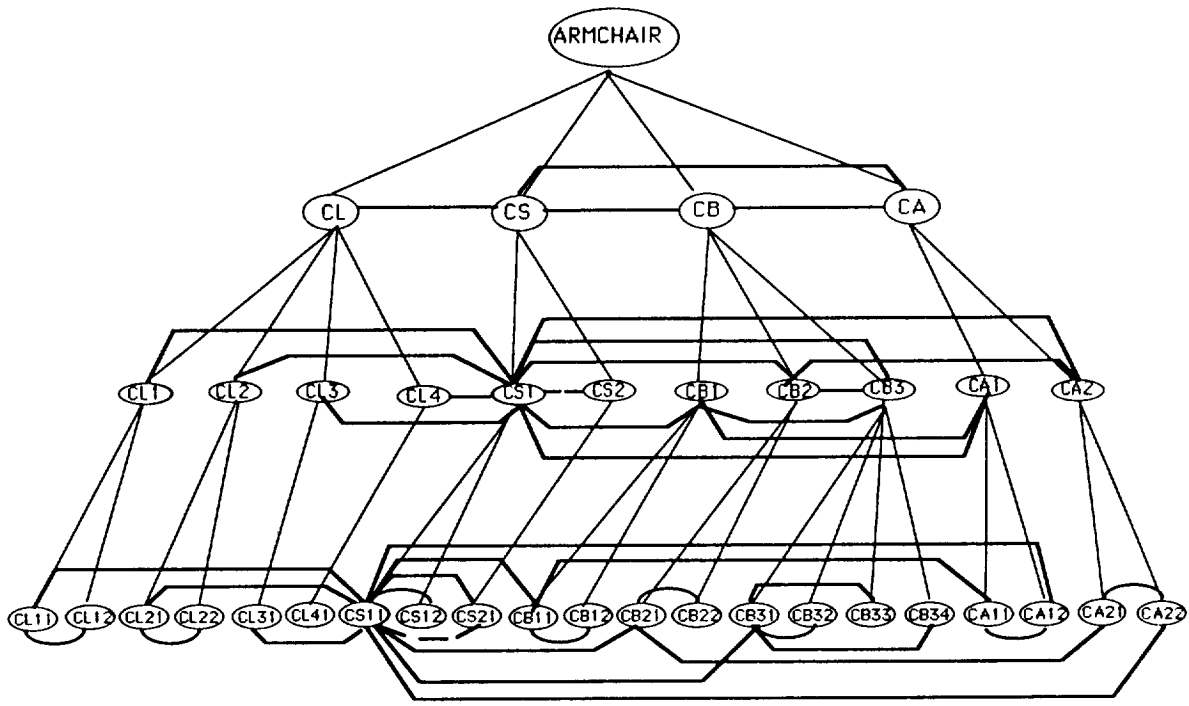


Figure 8. Graph of the Scope of Control (Assembly)

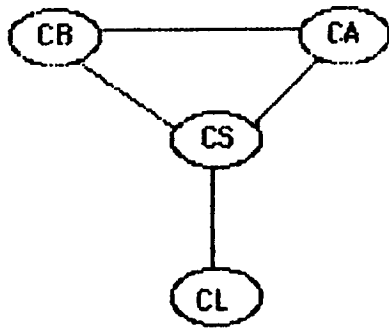


Figure 9. Tessellatum of the LR- level of the Control Scope

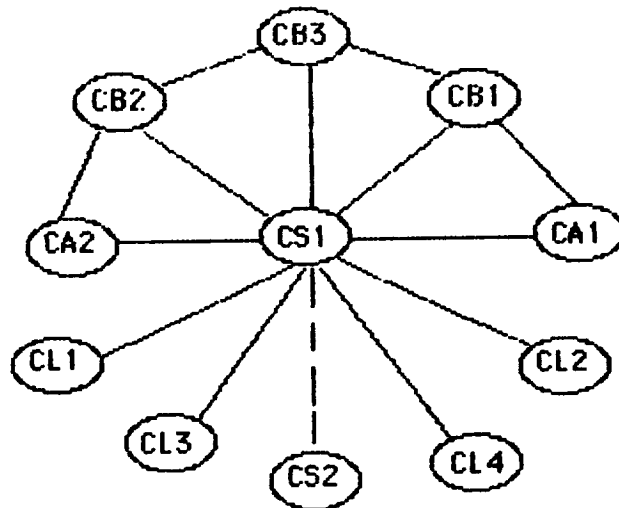


Figure 10. Tessellatum of the HR- level of the Control Scope





## PERCEPTUAL TELEROBOTICS

Panos A. Ligomenides

Cybernetics Research Laboratory, Electrical Engineering  
Department University of Maryland, College Park, Maryland 20742  
and Caelum Research Corp., Silver Spring, Maryland 20901

**Abstract**

A sensory world modeling system, congruent with a human expert's perception, is proposed. The **Experiential Knowledge Base (E\*KB)** system can provide a highly intelligible communication interface for telemonitoring and telecontrol of a real time robotic system operating in space. Paradigmatic acquisition of empirical perceptual knowledge, and real time experiential pattern recognition and knowledge integration are reviewed [1-5]. The cellular architecture and operation of the E\*KB system are also examined.

**1. Introduction**

Intelligent robotic decision making is a dynamic interplay of continually unfolding processes, characterized by approximate reasoning, fuzziness and dynamic readjustment. Problem definition and determination of goals, criteria and conditions, world modeling, planning and learning, are all contingent on perpetual gathering, processing and interpreting of information. In these activities, **knowledge** appears as a precious commodity, whose access, refinement and use can extend the ability for rational reasoning and problem solving.

Real time robotic decision making, totally automated or human-guided, depends critically on the availability of **comprehensible** internal models of the decision making world. Effective operation of adaptive robotic systems requires comprehensible man-machine communication with respect to cognition and command languages. Intelligible perceptual descriptions of the sensory world, based on interactive modeling of a human-expert's perception of it, would strengthen rational decision making.

The sensory world's meaningful structure and activity is evidenced by the underconstrained and often indeterminate **order** embedded in the sensory data. From the incredible richness and complexity of order which is filtered through the limited sensory bandwidths of the physical transducers, meaningful structural, temporal and causal embedded regularities are discerned and interpreted by imposing relevant **empirical elastic templates** of characteristic spatio-temporal **norms** of reference, which have been formed in the course of prolonged and active experience. The relevancy of the elastic templates used at any given instant is determined by the temporally shifting concerns, attention and attitudes of the decision maker. Interactively derived empirical

elastic templates for given  $k$ -norms,  $k \in K$ , and their subsequent application for real time perceptual recognition and internal modeling, are the pursuits of **Experiential Knowledge Engineering**. We call **experiential** the inferential capture of knowledge by direct associative ordering of sensory data, based on elastic templates which model a human expert's perception of the sensory world. We refer to the empirical elastic templates as "formal description schema" - models of the  $k$ -norm, denoted as " $fds_k$ " - models [1-5].

Human perceptual faculties of mapping discerned sensory order into meaningful predicates of probability, possibility or belief, function in a much more qualitative manner than one might deduce from examining most of the vision research literature, which is largely based on the computational framework of optical deconvolution, suggested by Marr, Barrow and Tenenbaum [6,7]. Logic-like deductive reasoning about invariable perceptual properties, based on previously established semantics of cognition, seem to follow better the biological paradigm [8,9]. The  $fds_k$ -model is a consistent, robust and computationally efficient tolerance model [1-5]. It assesses quantitative **conformity** to the  $k$ -norm, which is also expressible with a linguistic label (name) of **resemblance** to a norm. It functions procedurally and progressively on complete or partial data. The assessment is compatible with that of the human-expert whose perception it reflects.

In this paper we propose a telerobotic monitoring and control system, which is based on pattern recognition and internal modeling procedures that perform congruently to the perception of a human expert. This approach to telemonitoring and telecontrol aims in securing a most comprehensible communication interface between the robotic system and the human operator. It is based on modeling the empirical perceptual knowledge of a human expert in a highly interactive empirical knowledge acquisition system. The pattern recognition approach used is "paradigmatic" [9], as explained in section 2. The cellular architecture and operation of a real time recognition and world modeling system, the "Experiential Knowledge Base (E\*KB)" system [1-5, and references therein], functioning within the framework of a telerobotic space application as a natural extension of a decision maker's organic abilities, is also examined in section 3. Remarks follow in the concluding section.

## 2. Paradigmatic Pattern Recognition

### 2.1 Paradigmatic acquisition of empirical perceptual knowledge.

Pattern recognition is, in essence, an act of inductive inference. It is either the act of **re-cognition**, i.e. of "seeing something<sub>1</sub> as something<sub>2</sub>" [11], or the act of **cognition**, i.e. of forming a class by clustering items with some identifiable common properties. Both cases are now accepted as instances of **recognition** and are based on performing inductive inference.

Inductive inference, is a process of producing a general proposition based on a limited number of particular propositions, such that these become special instances of the former (a somewhat inexact remark, which however gives a general perspective of our argument). Going from particulars to universal, from concrete to abstract, is also the basic inductive function of paradigmatic pattern recognition. In a broad sense, this aspect of induction, which includes the advancement of argument by analogy, is considered to be the secret of the creative process in science.

The execution of induction involves certain creative operations, such as the creation of new hypotheses (abduction) or the identification of characteristic features (percepts) of an object, which are **extralogical**. They are acknowledged to be outside the domain of logical operations, and obviously outside the capability of the digital computer as we have come to know it. They, also, are value-dependent and much like the processes that underline creation in art.

A recognizable pattern is an object which is assessed to be a member of a class. For this reason, a pattern often plays a double role of an individual object and of a class. Logicians would define the concept associated with a pattern, or its class, by its intension or its extension. The intension is the collection of predicates which just defines the concept. The extension is the collection of individual objects which correspond to the concept and make up the class. In other words, the extension is the collection of all the objects that satisfy the predicates included in the intension. However, in real life situations, we "define" classes by class-samples (**paradigms** [10]). Shown a few samples of a "cat", a small child becomes capable of deciding whether any other animal is a cat or not. The child is not given the intension or extension of the class of cats and he seldom makes a mistake, although he may never be capable of defining the intension or extension of the class of cats. Theoretically speaking this is not a definition of a class. Nevertheless, the procedure generates in the human mind the capability of distinguishing a member from a nonmember of a class [10]. Excluding clustering, this procedure describes what we understand as pattern recognition: having been shown a few paradigms (and perhaps a few negative paradigms) of a class, one becomes capable of telling whether or not a new pattern belongs in this class. Together with the ability of clustering, this procedure underlines the schemata of human faculties of inferential capture of classified knowledge, i.e. of perceptual pattern recognition.

In paradigmatic pattern recognition which is based on interactive modeling of human perception, the class-defining elastic templates (the  $fds_k$ -models) are derived neither by intension nor by extension. They are derived only by paradigms. If the computer is ever to become capable of doing this job, it has to derive the class-defining features from the paradigms presented to it, so that the task of recognition becomes one of

intensional sorting and assessment of conformity, which the computer is capable of doing. But, the derivation of the class-defining properties from paradigms, is not a machine-feasible operation without human aid, as it is based on extralogical processes. Also, the introduction into the machine of a scale of "distance" for the assessment of conformity/resemblance between objects, can not be done meaningfully without telling the machine our value-judgments (often including aesthetic, moral, etc, judgements), which are of an entirely extralogical nature.

Since the inductive faculty can not be formulated as a necessary logical law, its implementation in a computer is often done by formulating it (by human intervention again) as some kind of heuristic principle, like the principle of minimum entropy. In the case of paradigmatic recognition, based on modeling the perception of some specific human expert, where "attention" may be varied at will, we have sought the aid of a human "expert" to identify and value the percepts of a reference norm, and to provide measures and methods for the assessment of conformity/resemblance. The interactive procedure for the derivation of the  $fds_k$ -models, and their application in pattern recognition, are briefly reviewed below. For more information, the reader is referred to [1-5], or directly to the author.

Samples of norm-patterns and of patterns with perceivable deviations from full conformity to the norm are inputted externally (as they are available), or they are generated by the modeling system. The human expert is called to verify discerned percepts and to assess conformity to the norm of the discerned perceptual organizations. The procedure is highly interactive and gradually leads to embedding the human expert's perceptual knowledge in a  $fds_k$ -model. The  $fds$ -modeling procedure is composed of three-phases. Early in the interactive session, in conjunction with noise treatment and abstraction operations, the "structural identity" of the  $k$ -norm,  $SkID$ , is established, to serve as an object locator and as an early perception device. The determination of the norm's percepts and evaluation of the elastic parameters for a full-conformity template, the  $M^k$ -model, is done in the second phase. Finally, the determination of conformity assessment procedures and measures and the derivation of the complete  $fds_k$ -model, are accomplished in the third phase. The  $fds_k$ -modeling procedure is outlined in figure 1.

## 2.2 Real time experiential recognition.

The real time application of the  $fds_k$ -model by programming a computer, in situations where data is sampled sequentially, is illustrated in figure 2. The sensory data is sampled and inputted serially, value-at-a-time:  $(Attr_p(Obj_q); value_y; @x, t; X, T)$ . In this case the recognition system accumulates an advancing window of sensory data, and it repeatedly tests for: (a) conformity to boundary or initial conditions, or (b) Conformity to structural identity ( $SkID$ ), before the abstraction-sensitive perceptual overlay ( $P_k$ ) assessment. In case (a) a window of a few successive samples allows advancing tests for boundary or initial conditions. Note that although

such conditions may be computed and tested progressively using only two advancing data-samples, more samples are included in the advancing test window, in order to test and establish the boundary or initial conditions in the presence of noise and measuring errors. The size of the sampling window depends on the sampling frequency and on the estimated noise and distortion levels. In case (b), the sampling window is stretched enough to allow for a delayed SkID test.

In situations where data may be generated parallelly by a sensory "retina", a neural pipeline may be used to implement the  $fds_k$ -model, as shown in figure 3. The  $fds_k$ -model includes a great deal of parallel operations of data processing and parameter assessment. This fact renders this pattern recognition model suitable for highly parallel implementation. Its inductive operations of order discernment and conformity assessment also make it suitable for realization with neural networks. The illustrated scheme for neural pipelined implementation, applied to 1D retinas, is currently being analysed in the Cybernetics Research Laboratory. An added objective in this endeavor is to generate real time temporal descriptions of scene events with high temporal resolution and small delay.

### 2.3 Experiential knowledge abstraction and integration.

The task of pattern recognition is to discern "lawful" order embedded in sensory data, and to interpret it by assessing conformity to characteristic constraints (percepts) of a norm. The temporal or spatial patterns of some measurable attribute of an object,  $Attr(Obj)$ , are classified in accordance to their assessed conformity/resemblance to various reference norms, selected for their relevance. In this process, the role of **abstraction** is fundamental. We recognize physical objects by recognizing the patterns of their attributes (selected for relevance) at some level of abstraction, which provides for the depth of detail required for the decision making task at hand. The procedural and progressive assessment of conformity to percepts of various norms by the corresponding  $fds_k$ -models is applied hierarchically to the objects, the attention scenes and to the activities at progressive levels of abstraction of the decision making world. Rule-based abstraction and integration procedures are also used in the implementation of  $fds_k$ -models, and for discernment and interpretation of causal relationships and other types of higher complexity order. The use of rule-based abstraction is merited because of the nature of the experiential knowledge, which is ill-structured, modular, often ambiguous and declarative. Such knowledge makes it difficult to represent it and process it with semantic networks, using labeled arcs and nodes.

Experiential knowledge is accumulated in real time in timed push-down buffers (FIFOs), weighted with "forgetting" coefficients. The short memory stacks function within "object-cells" in a cellular organization illustrated in figure 4. Asynchronous abstractions of temporal and spatial knowledge is performed in response to requests from the decision maker.

Attention worlds defined by the decision maker specify portions of the decision making world for which experiential internal models are requested. The requests may be about experiential knowledge referring to recent past, present and anticipated world scenes and activities. Hierarchically functioning processors perform a great variety of concurrent and distributed I/O and processing tasks, which implement pattern recognition, abstraction and integration operations.

Both vertical and horizontal abstraction is performed in the concurrent and distributed cellular architecture shown in figure 4. Attribute or object descriptions of higher complexity and composition are developed in real time along the vertical abstraction hierarchies. Selective cross-cell integration of experiential knowledge is performed along the horizontal abstraction hierarchy.

### **3. The Cellular E\*KB System as a Cognitive Prosthesis**

A telerobotic decision making environment for the E\*KB system, including other decision support components, communication and distribution network interfaces, is illustrated in figure 5. The decision maker performs tasks of diagnosis, planning and subcommand dissemination, being supported by Data Bases, by consultation Expert Knowledge Bases that contain various kinds of expertise, and by the Experiential Knowledge Base which provides a comprehensible man-machine interface through the human perceptual channel. The E\*KB system acts as a cognitive prosthesis to the decision maker's organic abilities for recognition and internal modeling of the sensory world. The E\*KB system [1-5] includes the NOESIS interactive subsystem for on-line acquisition of empirical knowledge, the OMIOSIS subsystem for capturing and labeling experiential knowledge embedded in sensory data, and the ICON subsystem for abstraction and integration of experiential knowledge in response to attention subworld specifications. Other components of experiential knowledge processing, such as cause-effect prediction and stability processors, are also included in the E\*KB system [1-5]. Research work on these subjects, relating to the development of an integrated E\*KB system, is currently conducted in the Cybernetics Research Laboratory and Caelum Research Corporation.

### **4. Remarks**

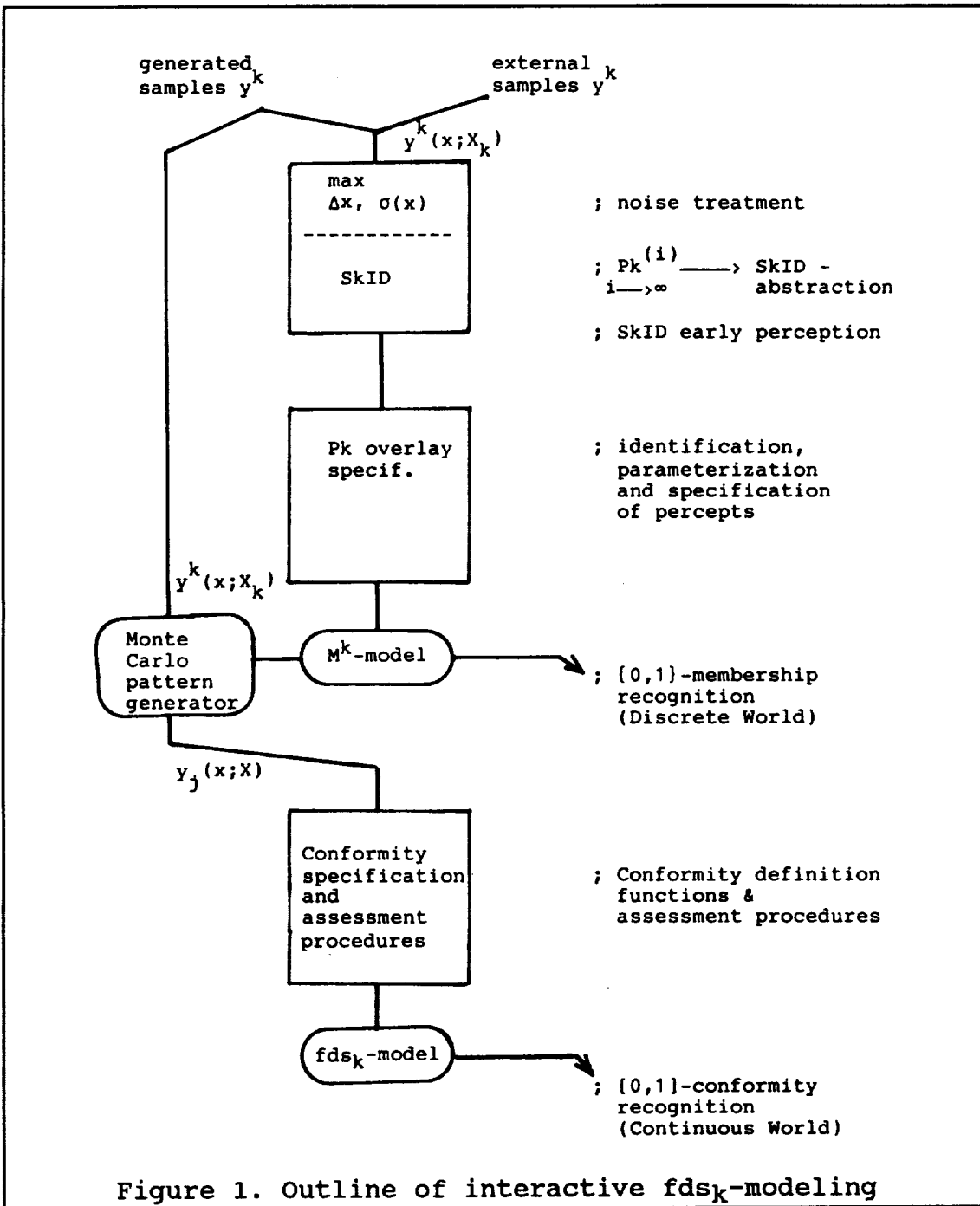
Reference norms are selected during our long experience guided by relevance-driven acts. When we find some relevance-based resemblance among various objects, which are perceived in connection with some decision making task, we apply a common label to them and we classify them in a "norm" class, forgoing on their many differences. Resemblance, contiguity in time or in space and cause-effect relationships are the kinds of perceptual "order" that enter the process of norm-class formation. In this paper, we have dealt only with the kind of order related to resemblance.

The commonsensical view of "classes" is that of a collection of "similar" objects. We might say that after centuries of scholastic detour, philosophers have come back to this simple-minded yet robust commonsensical view. From the theory of "general concepts" [Hume, Watanabe,10], which is based on the concept of similarity, we deal here with those aspects of similarity which have to do with "similarity of form" and partial conformity to form-relating characteristic perceptual constraints. We refer to this kind of similarity as "resemblance". In humans, it is a product of association formed by mental processes (Hume), which we model with our procedural elastic templates we call "formal description schemata".

Within the limited space available, we have attempted to present the conceptual and the architectural make up of the Experiential Knowledge Base system, which can serve as a cognitive prosthesis and as a comprehensible communication interface in a perceptual telerobotic application.

## References

1. P.A.Ligomenides, "The Experiential Knowledge Base as a Cognitive Prosthesis", in VISUAL LANGUAGES, Plenum 1986
2. P.A.Ligomenides, "Modeling Uncertainty in Human Perception", in UNCERTAINTY IN KNOWLEDGE BASED SYSTEMS, Springer Verlag 1987
3. P.A.Ligomenides, "Modeling Experiential Knowledge with Procedural Schemata", in UNCERTAINTY AND INTELLIGENT SYSTEMS, Springer-Verlag 1988
4. P.A.Ligomenides, "Real-Time Capture of Experiential Knowledge", IEEE Trans. Systems, Man and Cybernetics, SMC-18,4 1988
5. P.A.Ligomenides, "Capture of Experiential Knowledge by Conformity Assessment", invited lecture, 3rd Int'l Symp. on Knowledge Engineering, Oct.17-21, 1988, Madrid, Spain
6. D.Marr and T.Poggio, "Theory of Human Stereopsis", J. Opt.Soc. Amer., 67:1400 1977
7. D.Marr, "Visual Information Processing", in RECOGNITION OF PATTERN AND FORM, Springer-Verlag 1982
8. A.P.Pentland, "Towards a More Rational View of Logic", AI Magazine, 4, 1983
9. A.P.Pentland, (Ed), FROM PIXELS TO PREDICATES, Ablex 1986
10. S.Watanabe, PATTERN RECOGNITION: Human and Mechanical, Wiley, New York, 1985
11. L.Wittgenstein, PHILOSOPHICAL INVESTIGATION, Basil, Blackwell and Mott, Oxford, 1953





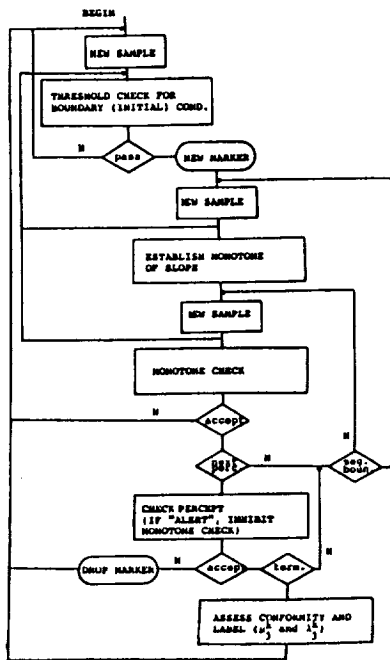


Figure 2. Sequential pattern recognition with  $fds_k$  models

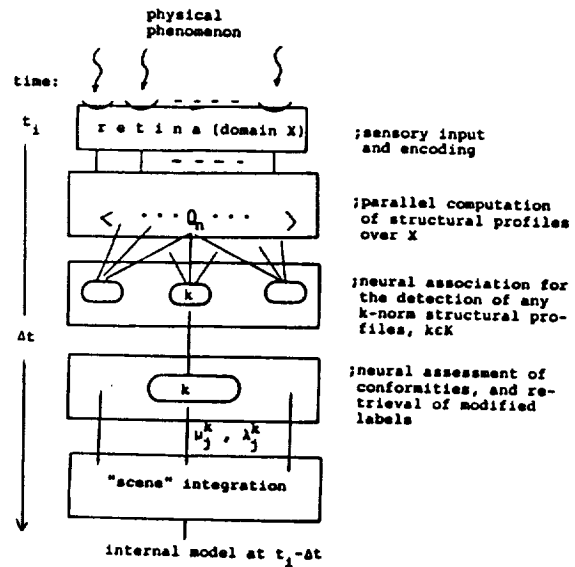


Figure 3. Parallel  $fds_k$  recognition with neural pipeline

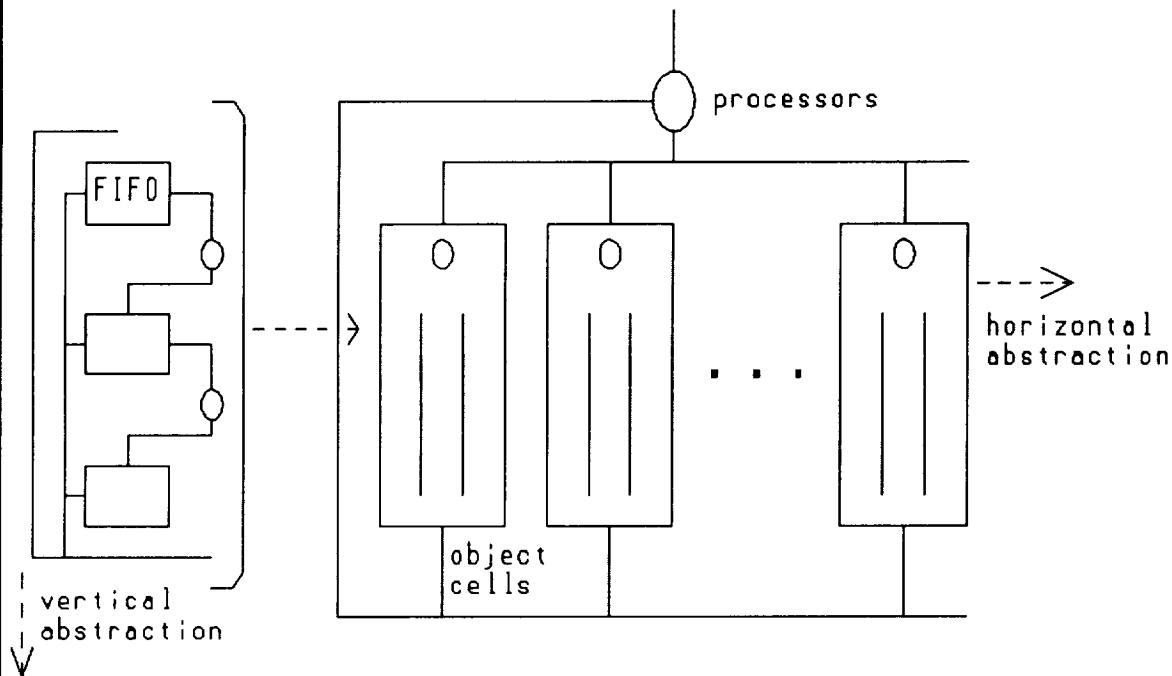
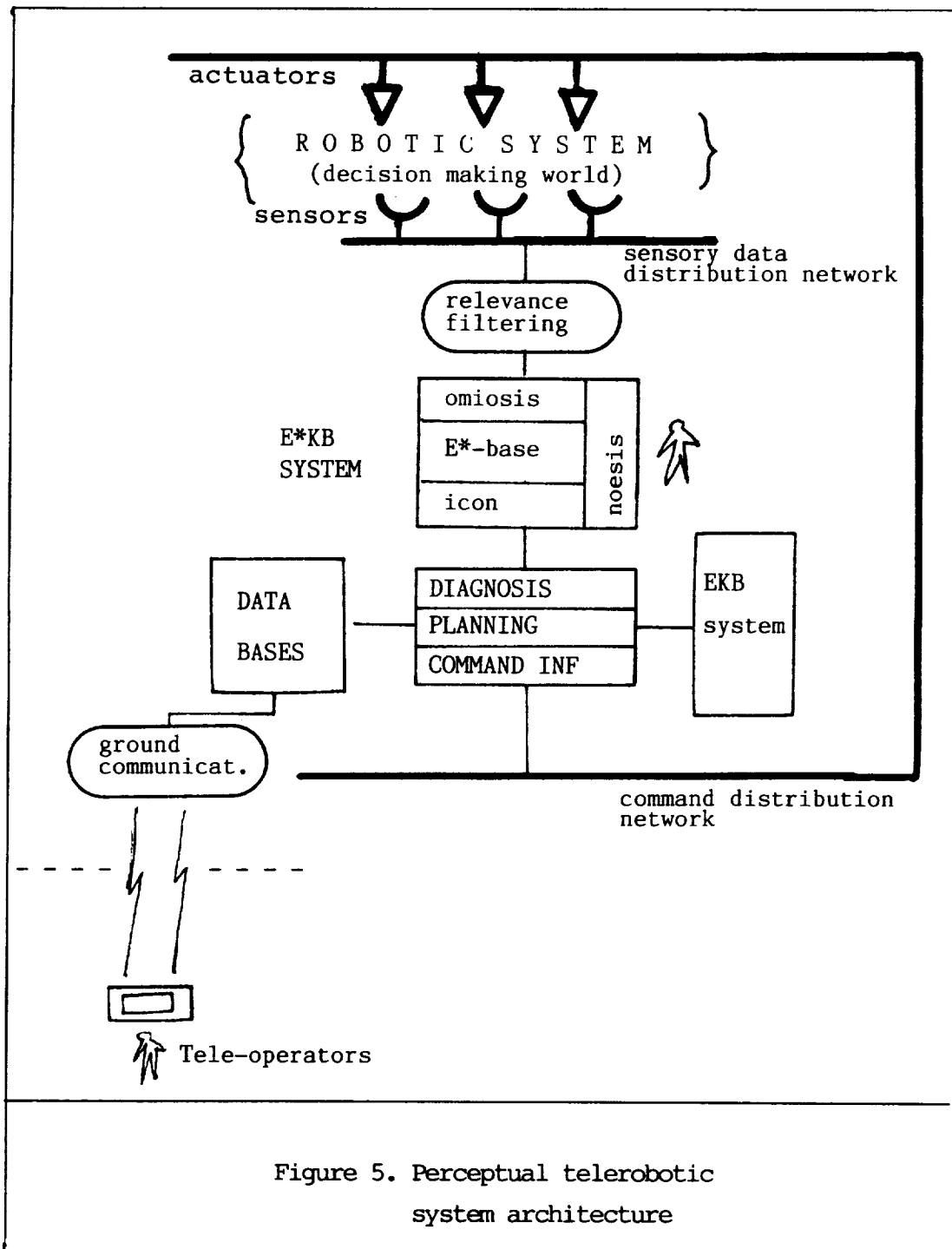


Figure 4. E\*KB cellular architecture



## Building an Environment Model Using Depth Information\*

Y. Roth-Tabak and Ramesh Jain

Artificial Intelligence Lab

Electrical Engineering and Computer Science Department

The University of Michigan, Ann Arbor MI 48109

### Abstract

Modeling the environment is one of the most crucial issues for the development and research of autonomous robot and tele-perception. Though the "physical robot" operates (navigates and performs various tasks) in the *real world*, any type of reasoning, such as situation assessment, planning or reasoning about action, is performed based on information in its *internal world*. Hence, the robot's intentional actions are inherently constrained by the models it has. These models may serve as interfaces between sensing modules and reasoning modules, or in the case of telerobots serve as interface between the human operator and the distant robot.

A robot operating in a known restricted environment may have *a priori* knowledge of its whole possible work domain, which will be assimilated in its *World Model*. As the information in the *World Model* is relatively fixed, an *Environment Model* must be introduced to cope with the changes in the environment and to allow exploring entirely new domains.

We introduce here an algorithm that uses dense range data collected at various positions in the environment to refine and update or generate a 3-D volumetric model of an environment. Our model which is intended for autonomous robot navigation and tele-perception, consists of cubic voxels with the possible attributes: Void, Full, and Unknown. We present experimental results from simulations of range data in synthetic environments. The quality of the results show great promise for dealing with noisy input data. The performance measures for the algorithm are defined, and quantitative results for noisy data and positional uncertainty are presented.

## 1 Introduction

Modeling the environment is one of the most crucial issues for intelligent autonomous system development and research. Though a "physical system" operates (navigates and performs various interactive tasks) in the *real world*, any type of reasoning, such as situation assessment, planning or reasoning about action, is performed based on information in its *internal world*. Hence the system's self and intentional action (non accidental or non human supervised) is inherently constrained by the internal models it has or may have of its environment and of the world. By the 'models a system may have' we mean the extent of the power of the representation scheme as opposed to a particular instance of a model which might be partial or incorrect.

The model used by such autonomous agents is referred to as *World Model* (WM), and it represents relatively fixed information about the world in which the system has to work. However, at any given time, only a small portion of the world model, called *environment* is used by an autonomous agent in its operation. The *Environment Model* (EM) at a given time instant should contain more detailed and explicit task-oriented information. The ultimate modeling scheme should consist of hierarchical decompositions on various scales such as a resolution scale and an abstraction scale. The resolution scale allows detailed (high resolution) inspection and reference of parts of the environment as well as a more general (low resolution) view. The abstraction scale, on which sensory data is on one side and a symbolic representation on the other, allows communication in both top-down and bottom up modes. Many researchers [5],[7], [10],[18],[1],[2],[3],[13], [21],[23],[17],[14],[26], have suggested various modeling schemes. In this work we concentrate on the volumetric level of the EM, on which information about *free* and *occupied* space

---

\*This work is sponsored in part by NASA Contract No. 958086

is represented explicitly. At this model level, updating operations can take place using raw sensory data, in case of range sensors, or processed data from any stereo or other depth recovery technique. In addition, although this type of model 'resides' on the very low level of the EM's abstraction hierarchy, it can be used directly by path planning and navigation modules, and object recognition and manipulation modules as well.

Herman and Kanade [10] reconstruct 3D scenes from a sequence of images where 3D wire frame descriptions help to construct surface based models. Chien and Aggarwal [6], Srivastava and Ahuja [25], and Potmesil [22] construct 3 dimensional object models from silhouettes from different views. Elfes [8], and Moravec [19] construct a 2 dimensional map of an environment using sonar readings and a Bayesian probabilistic approach to combine information from various sensor positions. Jain et al [11] use sparse range data provided by any stereo or other depth recovery technique and some worst-case assumption to construct a 3D model. All these techniques rely on relatively 'poor' data (external boundaries of objects, ultrasonic readings or sparse range data) to construct 2D or 3D maps of environments or objects, and hence need a relatively large number of views to obtain reliable models (relative to the complexity of the object or domain to be modeled), and usually require either some worst-case analysis or probabilistic analysis.

In the last few years range sensing technology has come a long way; faster, more accurate, and more reliable systems are now available [4]. In view of the nature of the information in the range images it is considered as a major source for 3-D model generation. As opposed to the methods described above, we consider a dense range sensor as the source of 'rich' information for our technique. In the work by Goldstein et al [9], dense range data was also used as a source for updating a world model, however, they were using a Combinatorial Geometry (CG) model (also known as Constructive Solid Geometry CSG) for describing objects in a scene, in particular they were using spheres as their initial building blocks. In our system we partition the space into a 3D matrix of cubic voxels, which imposes a different implementation and possible use. We introduce here an algorithm that uses dense range data from multiple viewpoints in an environment to refine a 3-D volumetric model of that environment. The locations and orientations of the sensor which are used for the updating process will be determined using reasoning, however, here the emphasis is on information assimilation. Our model is intended for autonomous robot navigation and tele-perception, and each voxel may have one of three attributes: Void, Full, or Unknown. The third attribute *Unknown* represents the voxels in space of which no information is known, and it can help guide an exploring module to locate a next good position for observation.

At the present we assume a static environment and the work described here follows this assumption, however, we intend to be able to relax this assumption and cope with dynamic environments using conflicting information between expected scene and a viewed scene. Such comparisons between expected and viewed scene may be also used to estimate the real position of the sensor.

## 1.1 Organization of the article

The following sections describe the method and some experimental results with simulated data. Section 2 describes the modeling scheme and the method to create and update the model. In this section a general review on World and Environment Models is introduced as well. In section 3 experimental results are described. These experiments include a generation of an environment volumetric model from several viewpoints, and a comparison with results obtained using noisy data and location uncertainty. In section 4 an evaluation of the results is provided followed by some conclusive remarks regarding the method's performance in different conditions. Section 5 describes possible extensions and directions for future work, and Section 6 provides a brief conclusion.

## 2 The Model and the Method

### 2.1 The model

We concentrate here on the lower abstraction levels of the EM, to demonstrate the use of sensory data to initiate a model. The model is a 3D volumetric grid of cubic voxels. The voxels are assigned three possible values: VOID - for empty voxels, FULL - for occupied voxels, and UNKNOWN for voxels for which no meaningful information has been obtained yet.

Other researchers such as [20] use different attributes in their grids and it might be claimed that the notion of Unknown may be captured by using uncertainties. We claim, however, that these notions are distinct and each

has its own importance in such models. These claims resemble claims of the Shafer-Dempster formalism [24] where Belief Functions are introduced to represent the "strength of the evidence" that specifically favors some proposition, in contrast to the Bayesian approach in which a unit of probability must be apportioned between the possible propositions.

*Uncertainty* provides information about the degree of confidence assigned to a certain piece of information about a voxel. The attribute *Unknown*, on the other hand, declares that there is no previous information available about a voxel. This attribute may be the key to a decision module searching for the next position for the sensor so as to capture as much as possible new or required information from a domain.

As opposed to [8],[19],[20] in our model we avoided assigning certainty levels to the attributes for the following reasons: 1. Range data produced by a dense range sensor is used, as opposed to ultrasonic sensors which usually have a 'wide' opening angle and impose more uncertainty on the location of the actual obstacles. 2. Dense range data is used which, as opposed to various stereo and depth recovery techniques from grey level images, provides a range reading for all the pixels in the image, and hence there are no 'spatial gaps' of depth information. 3. The updating technique is based on intersection with the previous model which speeds the operation as the model is being constructed; working with certainty levels would force us to scan the whole grid for every updating step. 4. This method treats the uncertainties and errors within itself, and provides a simple model, free of uncertainties, for use by other modules. However, we do not argue against using uncertainties, but merely point out to the fact that for the specific engineering task presented a model without uncertainty is sufficient.

## 2.2 Updating the Model

The model is initially entirely UNKNOWN. The position and orientation of the robot with respect to some general frame are assumed to be known for each view, however, some location uncertainty can be tolerated.

The operations performed include various transformations from a fixed Cartesian, coordinate system to a translated and rotated coordinate system, representing the sensor coordinate system. These transformations are used to determine the location of a point given in the global coordinate system in the sensor coordinate and vice-versa. Another required transformation is between the sensor Cartesian coordinate system and a sensor spherical coordinate system which represents the actual range image of the sensor.

A global Cartesian coordinate frame was defined as  $x, y$ , and  $z$ , where  $zx$  is the horizontal plane and  $y$  is the vertical axis (see figure 1).

The location of the sensor is defined as the coordinate point  $(z_0, x_0, y_0)$  in this global coordinate frame. The orientation of the sensor is specified in a sensor centered coordinate frame where  $z'$  is the viewing direction and  $y'$  is the vertical axis of the sensor's image plane.  $x'$  is defined so as to have a right-handed coordinate system (see figure 1).

The first transformation is between the global coordinate frame and the sensor centered Cartesian coordinate frame:

$$\begin{cases} x' = l_1(x - x_0) + m_1(y - y_0) + n_1(z - z_0) \\ y' = l_2(x - x_0) + m_2(y - y_0) + n_2(z - z_0) \\ z' = l_3(x - x_0) + m_3(y - y_0) + n_3(z - z_0) \end{cases}$$

and the inverse transformation:

$$\begin{cases} x = l_1x' + l_2y' + l_3z' + x_0 \\ y = m_1x' + m_2y' + m_3z' + y_0 \\ z = n_1x' + n_2y' + n_3z' + z_0 \end{cases}$$

Where  $l_1, m_1, n_1; l_2, m_2, n_2; l_3, m_3, n_3$ ; are the direction cosines of the  $x', y', z'$  axes relative to the  $x, y, z$  axes respectively, and  $(x_0, y_0, z_0)$  is the linear translation of the center of the coordinate system.

However, it is desirable to specify the transformations in terms of the *pan*, *tilt*, and *swing* angles, so that the transformation can be specified in terms of the orientation parameters explicitly available for the sensor.

Here are the direction cosines specified in terms of the *pan*, *tilt*, and *swing* angles where  $\alpha = \text{pan}$ ,  $\beta = \text{tilt}$ , and  $\gamma = \text{swing}$ :

$$\begin{aligned} l_1 &= \cos \alpha \cos \gamma - \sin \alpha \sin \beta \sin \gamma \\ l_2 &= -(\cos \alpha \sin \gamma + \sin \alpha \sin \beta \cos \gamma) \\ l_3 &= \sin \alpha \cos \beta \\ m_1 &= \cos \beta \sin \gamma \\ m_2 &= \cos \beta \cos \gamma \\ m_3 &= \sin \beta \\ n_1 &= -(\sin \alpha \cos \gamma + \cos \alpha \sin \beta \sin \gamma) \\ n_2 &= \sin \alpha \sin \gamma - \cos \alpha \sin \beta \cos \gamma \\ n_3 &= \cos \alpha \cos \beta \end{aligned}$$

In addition the following known transformation between the sensor centered Cartesian coordinate system to a spherical coordinate system, in which the sensor is actually defined, is used:

$$\begin{cases} x' = r \sin \theta \cos \phi \\ y' = r \sin \theta \sin \phi \\ z' = r \cos \theta \end{cases} \quad \begin{cases} r = \sqrt{(x')^2 + (y')^2 + (z')^2} \\ \theta = \cos^{-1}\left(\frac{z'}{r}\right) \\ \phi = \sin^{-1}\left(\frac{y'}{r \sin \theta}\right) \end{cases}$$

where  $\theta$  is the 'opening' angle, and  $\phi$  is the 'rotation' angle.

The Updating Algorithm can be described in words as following

1. Only voxels which are within the scope of the sensor are checked. By 'in the scope of the sensor' we mean both the 'angular' scope, i.e. being inside the cone in front of the sensor, and the 'distance' scope, i.e. being within the maximum range of the sensor.
2. Only voxels which are not yet VOID are checked (this implies an intersection with previous models).
3. For each of the voxels which are actually checked all of the 8 vertices are checked and compared to the actual pixel in the range image which points to their position in space.
4. If the maximum distance to any of the 8 vertices is smaller than the minimum range pointed by any of the range pixels, then that voxel is VOID.
5. If the 'range' of the vertices' distances intersects with the 'range' of the range pixels, and the difference between the maximum and the minimum range pixels is within a certain threshold, then a voxel is FULL.
6. Else it is unchanged.

This method has a few attributes worth mentioning. In step 3, the fact that 8 vertices are being checked has an inherent smoothing effect on the result. In most cases not all vertices will fall within the same range pixel (This depends on the size of the voxels and the resolution of the range image and can be guaranteed by controlling the size of the voxels), and hence noisy images to a certain extent will not have a strong impact on the result. In step 4, a certain threshold margin may be added to the above requirement in cases where there is some location uncertainty and its extent is known. This margin represents the worst case error which might result from such a location uncertainty. The threshold of Step 5 is introduced to avoid assigning FULL values to voxels which lie on or near sharp range discontinuities. This threshold is not critical since these voxels are not going to be assigned a VOID value, however it helps define the UNKNOWN regions in space and avoids assigning FULL values to the wrong voxels. Another quality of this method is that it is fully parallelizable in a straight forward manner.

### 3 Experimental Results

The experiments were performed with a 3D volumetric model of dimensions 64X64X16 voxels. However, there is no inherent restriction on the dimensions in the method.

To perform the experiments a synthetic domain was created (see figure 2) which is represented in a similar grid as the model is, with the attributes FULL and VOID only assigned to its voxels.

A range sensor which produces circular range images (see figure 3) was simulated. In the provided range images darker pixels represent closer range, and lighter color pixels represent further range. This range sensor has an opening cone of 60 spatial degrees (30 deg rotated around a symmetry axis). The sensor was described in spherical coordinates and the pixels were located by the 'opening' angle (distance from the center line of view) and the 'rotation' angle. This simulated sensor exhibited a resolution of 128X64 pixels, 128 on the rotation angle and 64 on the opening angle. The pixels values were integers between 0-128. Such a description of 'sensor' is not a common description for range sensors which usually define the parameters as elevation angle and azimuth angle [4]. However, the difference between the descriptions is only in the orientation of the spherical system ( $Z$  axis pointing to the viewing direction in the described sensor, as opposed to the  $Z$  axis being vertical to the viewing position). This type of description was chosen to produce higher resolution pixels in the center of an image and lower resolution at the image boundaries, and to produce a symmetric image form around its center line of view.. Nevertheless, the 'type' of range sensor (referring to the coordinate system in which the sensor is described) does not affect our method, but would merely require another transformation between the Cartesian coordinate system and the alternative spherical coordinate system. The sensor was 'positioned' in 12 different positions and orientations in the environment and the range images for these views were obtained (see table 1). The position is specified as

No. of View	z	x	y	pan (deg)	tilt (deg)	swing (deg)
1	56	6	10	180	-3	0
2	56	6	10	125	-3	0
3	56	6	10	100	-3	0
4	56	24	10	180	-3	0
5	56	52	10	100	-3	0
6	56	52	10	170	-3	0
7	36	46	10	235	-3	0
8	20	60	10	-20	-3	0
9	18	60	10	-30	-3	0
10	16	16	10	90	-3	0
11	5	28	10	-85	-3	0
12	8	8	10	60	-3	0

Table 1: The positions and orientations of the sensor used in the experiments

a 3-tuple coordinate ( $x,y,z$ ) relative to a fixed global coordinate frame, and the orientation is specified as a 3-tuple (pan, tilt, swing), where pan specifies the azimuth, tilt is the vertical direction where 0 deg is parallel to the ground, and swing is a rotation angle around the center line of view (this degree of freedom is seldom used nevertheless it is specified to have all the degrees of freedom available). The model which was initially UNKNOWN was then updated using these range images.

Experiments were also performed with the same data with added noise and with some location uncertainty. The noise that was added was a pseudo-Gaussian noise with  $\sigma$  (which is specified in the results) as the square root of the variance. When location uncertainty was added to the experiment, the same images were used but the algorithm used wrong position information. This position was produced by a pseudo-uniform random generator on the range 0-radius (The diameter is specified as a parameter in the results) from the actual position, and a pseudo-uniform random generator for the angle from the actual position on the range 0-2 $\Pi$ .

To evaluate the results numerically three parameters were introduced: *quality level*, *acquaintance level*, and *error level*. In the description of these parameters The 'EM' refers to the volumetric model that was generated using the Updating algorithm, and 'The No. of voxels' referenced in the denominator of the following expressions refers refers to the number of the (FULL or VOID) voxels in the real domain.

$$\text{quality level} = \frac{\text{No. of correct VOID voxels in the EM}}{\text{Total No. of VOID voxels}}$$

$$\text{acquaintance level} = 1 - \frac{\text{No. of UNKN EM voxels VOID in real domain}}{\text{Total No. of VOID voxels}}$$

$$\text{error level} = \frac{\text{No. of wrong VOID voxels in the EM}}{\text{Total No. of FULL voxels}}$$

All these parameters are presented in percentile. The *quality level* represents the percentage of the free space which was correctly found. For robot navigation and path planning this is an important aspect for evaluating the method. Since the goal is to correctly identify the clear passages in a domain, the percentage of the VOID voxels that were found gives a quantitative estimate of the quality of the model. The *acquaintance level* is 100 – (minus) the percentage of the UNKNOWN voxels which are actually VOID. This parameter represents the level of acquaintance the robot-model has with the environment and helps to evaluate the performance with regard to the number of views taken. The *error level* represents the percentage of wrong VOID voxels to total number of FULL voxels. This parameter actually specifies the level of confidence we can assign to the model's accuracy. The two parameters of quality and error combined are obviously the bottom line for evaluating the model.

The results for a lower resolution model were evaluated in the same manner too. The reason for this additional evaluation is that the above parameters do not specify the distribution of the wrong or correct voxels in the model. Moving to one lower level of resolution allows to evaluate the extent of the quality and of the errors in the model with regard to navigation and path planning in an additional perspective.

As for the time it takes to generate the model, since we were interested more in competence issues than in computational issues only rough run time measurements were performed, the results are still quite fast. The times provided here are real computer run time and not CPU time. We ran it on an Apollo DN4000 workstation, and for the first 3 views it took between 45-60 sec. to update the model, and then the time for updating the model drops down to 15-20 sec. for the last views. Considering the possibility of parallelism, since the algorithm is performed serially on all of the voxels, the increase in speed will be bound by the extent of the parallel machine. This means that any parallel machine can be used to its full potential with minor modifications to the algorithm.

## 4 Discussion

The results for added noise show that the method is not susceptible to noise. In full resolution up to noise levels of normal standard deviation  $\sigma = 3$  an error level up to 1% only is produced. Furthermore, at the lower resolution level the error level drops below 1% even for high noise levels of  $\sigma = 8$ . As the range image pixel values are between 0-128, such noise levels are enormous. This resistivity to noise was expected due to the natural smoothing which is performed implicitly (see table 2). As for location uncertainty, at full resolution only up to diameter = 3 produces acceptable errors. At the lower resolution up to diameter = 4 the error levels are still low, but then they rise sharply, and the effect of the lower resolution we had on the noise does not appear here (see table 3). As mentioned before, a threshold may be added to deal with location uncertainty. However, there will be a trade-off between the quality level and the error level when adding this threshold.

## 5 Future Work

The overall performance of this method shows a lot of promise for producing an adequate volumetric model of an environment as exhibited in the results presented here. We intend to pursue this research in the following directions:

- Investigate this method with real data. This will involve both adapting the transformations to the geometry of a specific real range sensor, and also collecting the actual data in a controlled environment where the results can be properly evaluated.
- Investigate additional methods for dealing with location uncertainty. We think that an adaptive technique which changes its mode of operation according to the level of acquaintance with the environment may provide the desirable results.



- Investigating possible optimizations of the actual algorithm. The algorithm which was used to generate the results presented here was written with the intent to check its competence for the task of updating a volumetric model. We would like to investigate it from computational aspects too.
- Investigate algorithms and methods for guiding an autonomous agent carrying the range sensor to a potentially informative new position and orientation. The idea is to use the UNKNOWN voxels as indicators for parts of the space which should still be explored.
- Investigate dynamic domains. This must involve additional higher level information such as attributes for static voxels, potentially mobile voxels, and actively moving voxels.

## 6 Conclusion

We presented here a method for generating, refining and updating a volumetric Environment Model of a domain using dense range data. Such a method may be used by an autonomous intelligent system for navigation and path planning, as well as for object recognition and manipulation, or for transferring spatial information from a remote sensing agent to its operator. Performance measures for the algorithm were defined, and quantitative results with noisy data and positional uncertainty were provided. Both the quality of the results, the stability of the method under noisy conditions, the relative speed of computation, and the real 3D quality of the information acquired, demonstrate the potential of the method for autonomous intelligent systems.

Though at the present we are assuming a static environment, we intend to be able to cope with dynamic environments using conflicting information between expected scene and viewed scene. Such comparisons between expected and actually viewed scene may be also used to estimate the real position of a sensor under location uncertainty conditions.

## References

- [1] K. M. Andress and A.C Kak, "A Production System Environment for Integrating Knowledge with Vision Data", *Proc. of the AAAI Sensor Fusion Workshop*, pp. 1-11, 1987.
- [2] Anthony Richard, "Spatial Reasoning Using an Object Oriented Spatial DBMS", *Proc. of the AAAI Sensor Fusion Workshop*, pp. 42-51, 1987.
- [3] Asada Minoru, "Building a 3-D World Model for a Mobile Robot from Sensory Data", *Report CAR-TR-332 CS-TR-1936 University of Maryland*, October 1987.
- [4] Paul J. Besl, "Active, Optical Range Imaging Sensors", *Machine Vision and Application (1988) 1*, pp. 127-152 1988.
- [5] Rodney A. Brooks, "Visual Map Making for a Mobile Robot", *Proc IEEE Int. Conf. on Robotics and Automation*, pp.824-829 1985.
- [6] C.H. Chien and J.K. Aggarwal, "Identification of 3D Objects from Multiple Silhouettes Using Quadrees / Octrees", *Computer Vision, Graphics, and Image Processing* 36, pp. 256-273 1986.
- [7] James L. Crowley, "Representation and Maintenance of a Composite Surface Model", *Proc. of the 1986 IEEE International Conf. on Robotics and Automation*, pp. 1455-1462, April 1986.
- [8] Alberto Elfes, "Sonar-Based Real-World Mapping and Navigation", *IEEE Journal of Robotics and Automation*, Vol. RA-3, No.3, pp 249-265, June 1987.
- [9] M. Goldstein, F.G. Pin, G. de Saussure, and C.R. Weisbin, "3-D World Modeling With Updating Capability Based on Combinatorial Geometry", *Proc. of the Workshop on Space Telerobotics*, pp. 273-282, 1987.
- [10] Martin Herman and Takeo Kanade, "Incremental Reconstruction of 3D scenes from Multiple, Complex Images", *Prager, USA*, 1981.

- [11] R. Jain, Y. Roth-Tabak and K. Skifstad, "Hyperpyramids for Vision-Driven Navigation", *Proc. AAI-VI (Orlando)*, April 1988.
- [12] Stephen Kaplan, and Rachel Kaplan, "Cognition and Environment, Functioning in an Uncertain World", *Proc. of the AAAI Workshop on Sensor Fusion*, pp. 390-404, 1987.
- [13] Benjamin J. Kuipers and Y. T. Byun, "A Qualitative Approach to Robot Exploration and Map-Learning", *Proc. of the AAAI Workshop on Sensor Fusion*, pp. 390-404, 1987.
- [14] Benjamin J. Kuipers and Tod S. Levitt, "Navigation and Mapping in Large Scale Space", *AI Magazine* Vol.9 No.2, pp. 25-43, 1988.
- [15] Daryl T. Lawton, Tod S. Levitt, Christopher C. McConnel, Philip C. Nelson, and Jay Glicksman, "Environmental Modeling and Recognition for an Autonomous Land Vehicle", *Proc. of the Workshop on Space Telerobotics*, pp.313-336, 1987.
- [16] Tod S. Levitt, Daryl T. Lawton, David M. Chelberg, Philip C. Nelson, and John W. Dye, "Visual Memory Structure for a Mobile Robot", *Proc. of the AAAI Sensor Fusion Workshop*, pp. 92-105, 1987.
- [17] Tod S. Levitt, Daryl T. Lawton, David M. Chelberg, Kerry V. Koitzsch, and John W. Dye, "Qualitative Navigation II", *Proc. of the DARPA Image Understanding Workshop*, pp. 319-326, 1988.
- [18] Herman Martin and Takeo Kanade, "Incremental Reconstruction of 3D Scenes from Multiple Complex Images", *Artificial Intelligence* 30 pp. 289-341, 1986.
- [19] H.P. Moravec, "Certainty Grids for Mobile Robots", *Proc. of the Workshop on Space Telerobotics*, pp. 307-312, 1987.
- [20] Hans P. Moravec, "Sensor Fusion in Certainty Grids for Mobile Robots", *AI Magazine* Vol.9 No.2, pp. 61-74, 1988.
- [21] I.J. Oppenheim, D.R. Rehak, W.T. Keirouz and R.F. Woodbury, "Robotic Task Planning, Domain Modeling and Geometric Reasoning", *EPRI NP-5525 Project 2515-1 Final Report*, December 1987.
- [22] Michael Potmesil, "Generating Octree Models of 3D Objects from Their Silhouettes in a Sequence of Images", *Computer Vision, Graphics, and Image Processing* 40, pp. 1-29 1987.
- [23] Nageswara S.V. Rao, S.S. Iyengar B. John Oommen and R. L. Kashyap, "On Terrain Model Acquisition by a Point Mobile Robot Amidst Polyhedral Obstacles", *IEEE Journal of Robotics and Automation*, Vol.4, No.4, pp. 283-288, August 1988.
- [24] Glen Shafer, "A Mathematical Theory of Evidence", *Princeton University Press*, 1976
- [25] Sanjay K. Srivastava and Narendra Ahuja, "An Algorithm for Generating Octrees From Object Silhouettes in Perspective Views", *Proc. of the IEEE Workshop on Computer Vision*, pp. 363-365, Nov 30 - Dec 2 1987.
- [26] Ellen L. Walker, Martin Herman, and Takeo Kanade, "A Framework for Representing and Reasoning about Three-Dimensional Objects for Vision", *AI Magazine* Vol.9 No.2, pp. 47-58, Summer 1988.
- [27] Wai K. Yeap, "Towards a Computational Theory of Cognitive Maps", *Artificial Intelligence* 34, pp. 297-360, 1988.

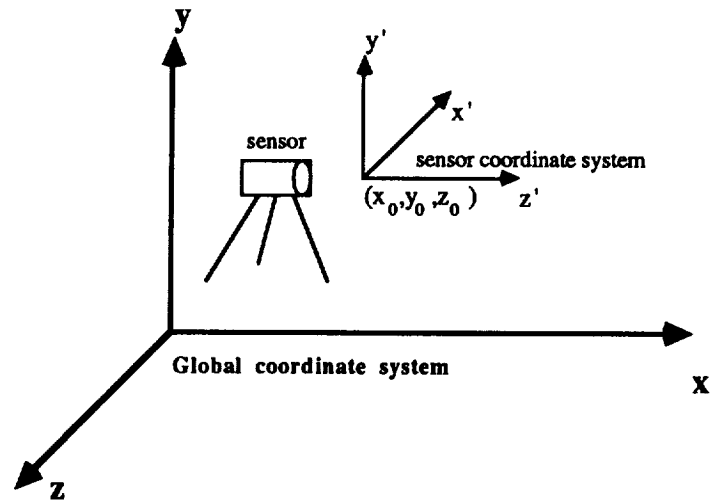


Figure 1: Transformation & rotation of Cartesian coordinates

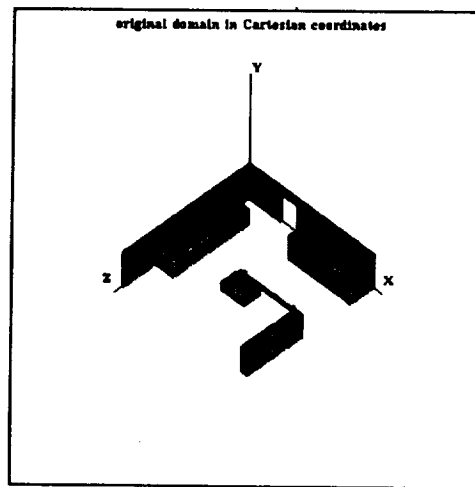


Figure 2: The synthetic domain

$\sigma$	Quality level %	Acquaintance level %	Error level %
0	94.7	99.9	0
1	94.5	99.9	0
2	93.2	99.9	0.07
3	92.2	99.9	0
4	91.5	99.9	0
5	90.9	99.9	0.07
6	90.1	99.8	0.13
7	89.8	99.5	0.13
8	89.3	98.9	0.53

Table 2: The effect of noisy range images on the model at a lower resolution

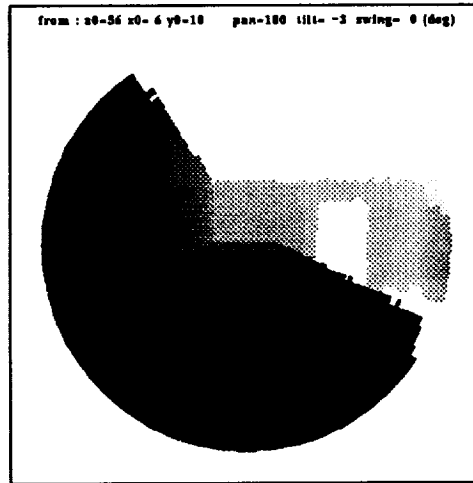


Figure 3: The synthetic range image from the first position

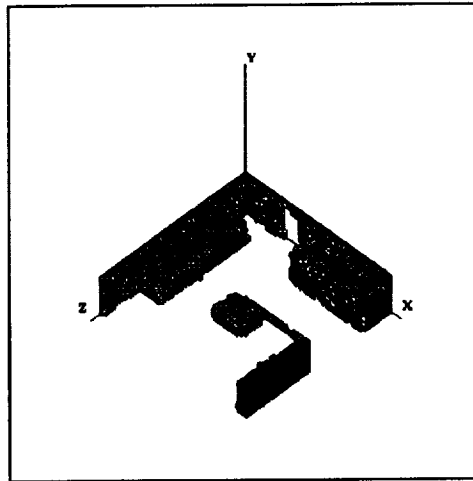


Figure 4: The model after 12 views

diameter of uncertainty	Quality level %	Acquaintance level %	Error level %
0	94.7	99.9	0
1	95.1	99.9	0
2	96.8	100.0	0
3	97.6	99.9	0.20
4	96.8	99.9	0.86
5	95.7	99.8	5.32
6	97.2	99.9	7.98
7	97.4	99.7	9.51
8	95.3	98.8	10.24

Table 3: The effect of location uncertainty on the model at a lower resolution

## **ROVERS**



## HERMIES-III: A STEP TOWARD AUTONOMOUS MOBILITY, MANIPULATION AND PERCEPTION

C. R. Weisbin, B. L. Burks, J. R. Einstein,  
R. R. Feezell, W. W. Manges and D. H. Thompson

Oak Ridge National Laboratory  
Robotics and Intelligent Systems Program  
Oak Ridge, TN 37831-6364

### ABSTRACT

HERMIES-III is an autonomous robot comprised of a seven degree-of-freedom (DOF) manipulator designed for human scale tasks, a laser range finder, a sonar array, an omnidirectional wheel-driven chassis, multiple cameras, and a dual computer system containing a 16-node hypercube expandable to 128 nodes. The current experimental program involves performance of human-scale tasks (e.g., valve manipulation, use of tools), integration of a dexterous manipulator and platform motion in geometrically complex environments, and effective use of multiple cooperating robots (HERMIES-IIB and HERMIES-III). The environment in which the robots operate has been designed to include multiple valves, pipes, meters, obstacles on the floor, valves occluded from view, and multiple paths of differing navigation complexity. The ongoing research program supports the development of autonomous capability for HERMIES-IIB and III to perform complex navigation and manipulation under time constraints, while dealing with imprecise sensory information.

### I. INTRODUCTION

The Center for Engineering Systems Advanced Research (CESAR) at the Oak Ridge National Laboratory (ORNL) focuses its research on the development and experimental validation of intelligent control techniques for autonomous, mobile robots able to plan and perform a variety of assigned tasks in unstructured environments.<sup>1</sup> The assignments originate with the human supervisors in a remote "control station," and the robot then performs detailed implementation planning and executes the tasks. Since the operational environment is generally dynamic, the robot must be in sensory contact with its surroundings to capture and recognize changes which bear on its task objectives and, if necessary, replan its behavior. These capabilities imply that the robot has cognitive capabilities that enable it to form and modify a model of the world around it and relate this world model causally to the task objectives. Research is also conducted to enable the robot to learn from its past experience, and thus improve its performance.

CESAR's principal current objectives are (a) to achieve a level of technological capability which would enable the autonomous performance of classes of navigation and manipulation tasks of human scale in a spatially complex environment; (b) to use these performance tasks as a focus for establishing and conducting its research objectives. In order to achieve these objectives, CESAR is developing a series of mobile autonomous

robot vehicles named HERMIES (Hostile Environment Robotic Machine Intelligence Experiment Series) as experimental test beds which enable validation of this research and demonstration of its results.<sup>2-4</sup> Our newest research robot, HERMIES-III,<sup>5,6</sup> includes the functional capabilities which permit research in combined mobility/manipulation, and allows us to experiment with cooperative control of multiple robots having different capabilities.

## II. ROBOT EVOLUTION TO HUMAN-SCALE EXPERIMENTS: HERMIES-III

Although HERMIES-IIB<sup>4</sup> is a powerful and versatile research tool, it has limited manipulative capabilities. In order to approach human-scale performance, CESAR has designed and is assembling a much larger test bed, HERMIES-III,<sup>5,6</sup> to be used in future experiments. This section briefly describes the hardware of HERMIES-III, a proposed software architecture, and a set of experiments which build upon those previously performed with HERMIES-IIB.

### A. HERMIES-III Hardware

HERMIES-III is a battery-powered robot currently under construction with operational availability in the spring of 1989 (Fig. 1). It is comprised of:

- A wheel-driven chassis ( $4' \times 5' \times 2'$ ) with omni-directional steering capability. Two steering wheels and four corner caster wheels are used to distribute the approximately 2700 pounds of vehicle weight over a  $10 \text{ ft}^2$  area. A pair of latitudinal and longitudinal hinges enable the vehicle to keep all 6 wheels on the ground even while traversing mildly uneven terrain.
- Initially one and later two manipulators; the current manipulator system comprises the CESAR Research Manipulator (CESARm)<sup>7,8</sup> which is a relatively high capacity-to-weight ( $\sim 1/10$ ) manipulator with 7-DOF and a spherical 3-DOF wrist. The arm now contains only a gripper and later will be augmented with a multi-fingered hand. CESARm is mounted about  $3\frac{1}{2}$  feet off the floor so that the end effector will reach from the floor to about 8 feet high, and 4 feet beyond the front edge of the vehicle. CESARm's characteristics are being benchmarked, and its control algorithms will be in the public domain;
- A sensor suite including an Odetics laser range camera, two pairs of CCD cameras, an array of 32 sonar transceivers on the chassis sides, and encoders or resolvers on motor shafts and manipulator joints. The laser range camera is mounted on a rotatable mount and provides range and reflectance data for  $128 \times 128$  ray directions within a  $60 \times 60$  degree field of view. All cameras are on pan and tilt platforms; additional pan and tilt platforms and mounting locations will be available for rapid addition of other sensors. Force-feedback and tactile sensors, and wrist mounted cameras for arm control will be mounted on CESARm in the near future;
- A dual computer system comprising IBM PC/AT-NCUBE and VME bus based systems with provisions for up to 128 NCUBE nodes and five Motorola 68020 processors. There



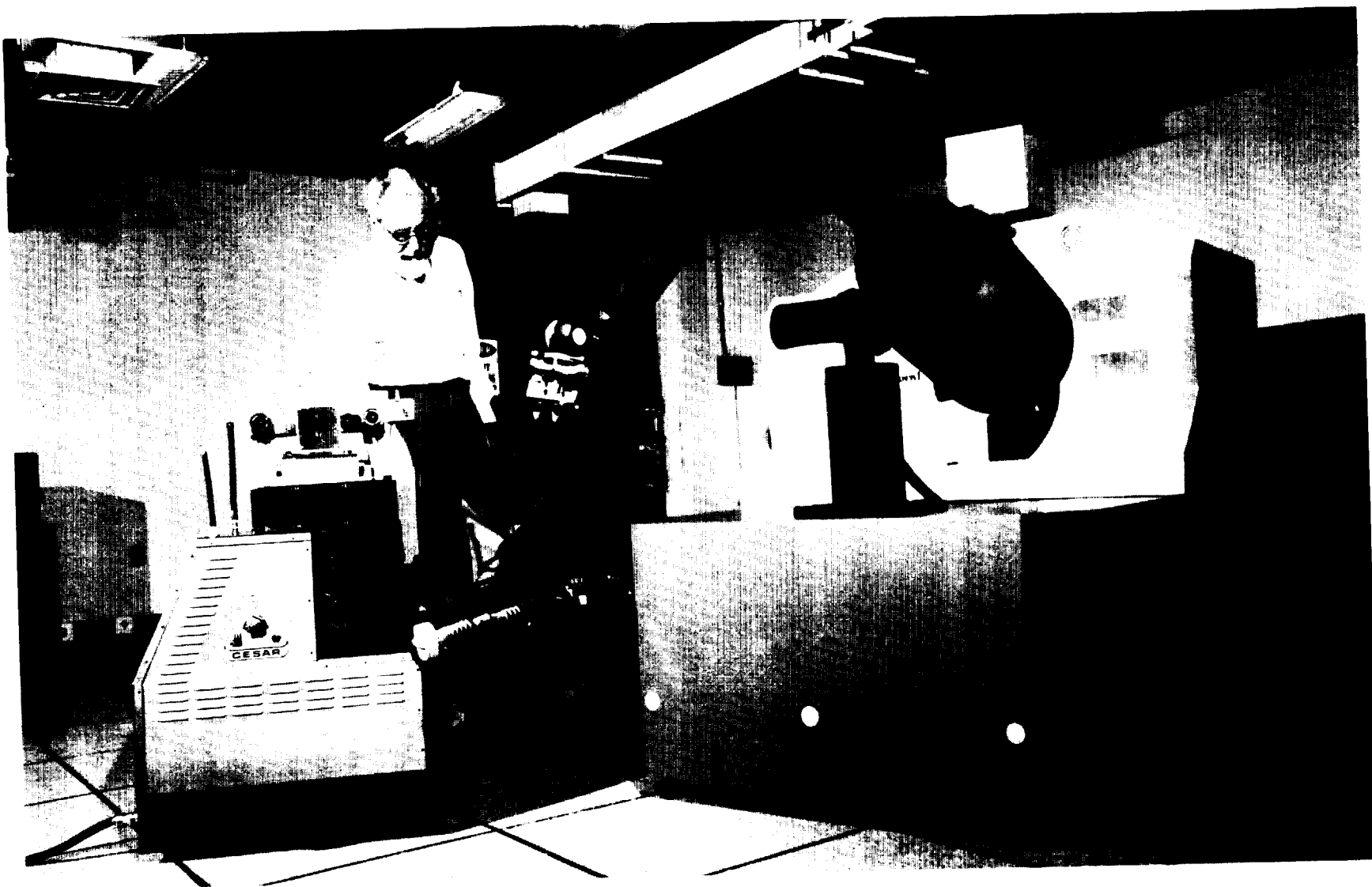


Fig. 1. The HERMIES-III mobile robot full-scale model is pictured alongside the HERMIES-IIB machine.

are 2 Mbytes of RAM and  $\frac{1}{2}$  Mbyte for each NCUBE node. Mass storage is provided by two 40 Mbyte hard disks and a 1.2 Mbyte floppy disk.

- An RS-232 wireless model used for communication with the on-board computer system during experiments. The on-board batteries allow 3 to 4 hours' normal operation of all components. The sizes of the platform, batteries, and electronics compartment allow later expansion, including the addition of a second arm, more sensors, and additions to the computer system.

- The specifications for the mechanical design are:

Total weight	1230 kg (2700 lbs.)
Batteries	410 kg (900 lbs.)
	48 V at 110 amp-hours
	24 V at 220 amp-hours
vehicle speed	60 cm/sec (2 ft/sec)
vehicle acceleration	120 cm/sec/sec (4 ft/sec/sec)
arm tip speed	300 cm/sec (10 ft/sec)
arm weight	160 kg (350 lbs.)
arm payload	15 kg (33 lbs.)
arm reach	137 cm (54 in.)

A symbolic layout of the hardware architecture is presented in Fig. 2.

## B. Proposed Software Architecture

HERMIES-III has been designed and constructed to provide significant hardware capability for perception, manipulation, mobility and computing; accordingly, the software for control of this vehicle will require a great degree of modularity, standardization, and hierarchy. Figure 3 represents our current view<sup>5</sup> of a suitable logical architecture for HERMIES-III. Before describing the diagram in more detail, three caveats are appropriate. First, this architecture is only now being implemented. The authors clearly recognize that experience will suggest revisions particularly for the data flow paths between modules. Second, the structure must accommodate and facilitate the implementation of a "brain" for HERMIES-III near-term demonstrations as well as a mechanism for the testing of basic research concepts. Demonstrations and basic research sometimes conflict in terms of requirements for standardization. Finally, the figure presents only a coarse look at the overall structure. The specific algorithms to be used in any given module and the data structures and interface specifications have not yet been finalized.

The envisioned structure includes five major components: Human Machine Interaction, Control, Mobility, Perception, and Manipulation. Although there is a Public Knowledge database, the architecture permits private "world" models to be maintained locally within any given task module. There is no single box allocated to Machine Learning. Our current view is that specification of a single learning program independent of local context and need would not be optimal; i.e., machine learning capability is subsumed within each of the modules according to need. The architecture is intended to accommodate a wide range of situations and tasks.

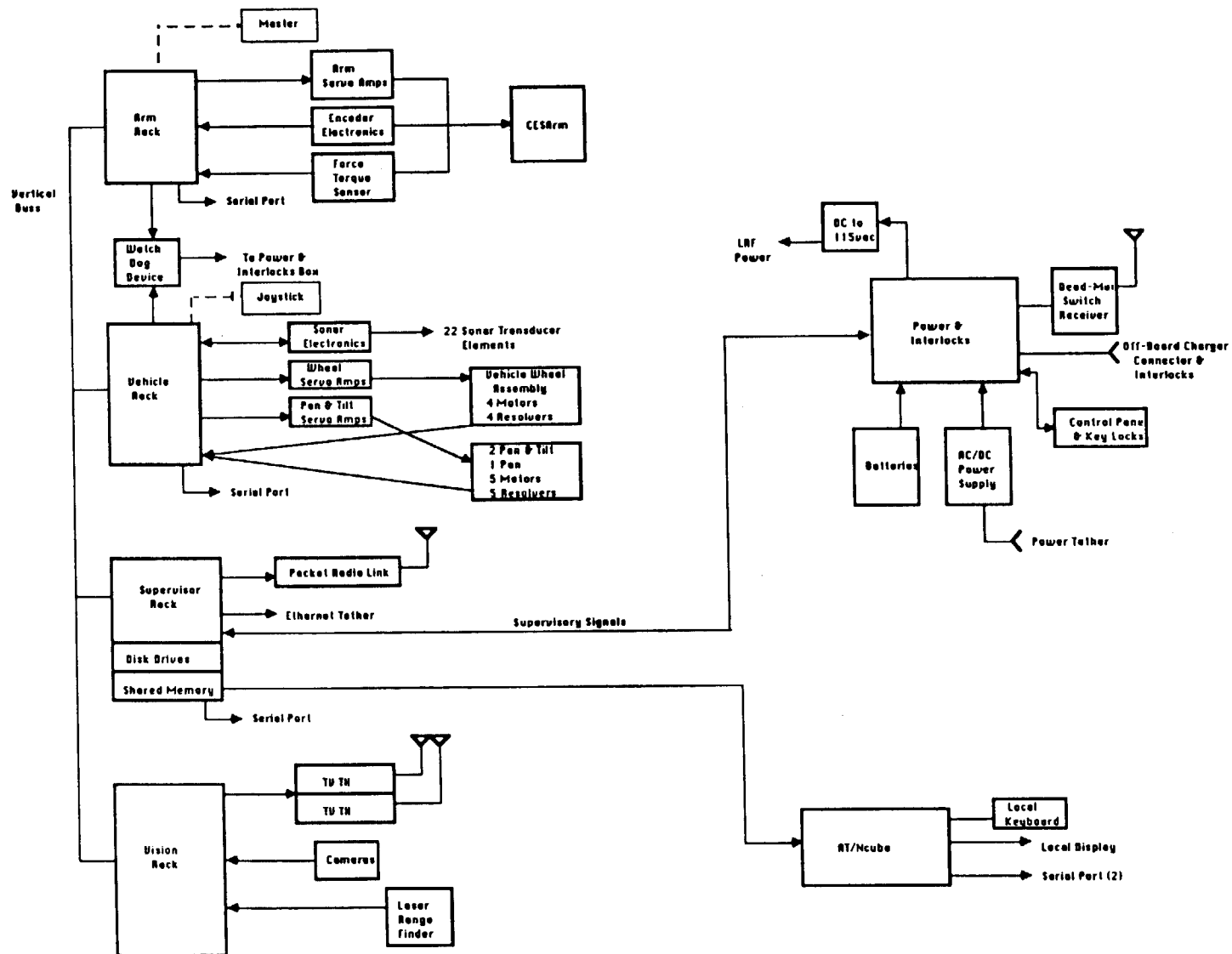


Fig. 2. The HERMIES-III initial hardware architecture.

We begin our discussion of Fig. 3 with the Human-Machine Interaction component. A goal is specified by the user through a suitable Human-Machine Interface which can be either keyboard or voice command. The next module, Job Decomposition Among Resources, is intended to divide the overall goal appropriately among the potential resources according to capability, availability, etc. For example, HERMIES-IIB might find, read, and monitor a control panel while HERMIES-III is attempting to suitably manipulate related valve(s).

Once the subgoals for each resource have been established a Task Planner is invoked to determine the steps needed to achieve that subgoal. At this level of planning, the tasks are phrased symbolically (e.g., avoid the obstacle). The symbolic tasks must be transformed to numerical procedures and ultimately to robot primitive operations (e.g., one turn of the wheel). These transformations are implemented through the Symbolic/Numeric Coordinator which mediates where appropriate between the Symbolic Task Planner and the more numerically intensive Mobility, Perception, and Manipulation modules. It should be noted that the overall software architecture is intended to allow for both vertical and horizontal communication. The vertical hierarchy represents task decomposition while the horizontal communications facilitate joint tasking (e.g., simultaneous arm and platform motion) and "reflex" action.

The Human-Machine Interaction, Task Planner, Mobility, Perception and Manipulation modules follow the three level Organization, Coordination and Execution hierarchy suggested by Saridis.<sup>9</sup> The addition of horizontal communication facilitates interactive tasks following the structure suggested by Albus.<sup>10</sup> The overall structure is intended to allow for decentralized control and asynchronous operation. The Mobility module considers global route planning at the highest level with increasing resolution in local navigation and obstacle avoidance at the lower levels. The Perception module integrates information from multiple sensors (e.g., sonar, vision, laser, force, tactile) at different levels with varying resolution following processing and interpretation of data from a single sensor. Similarly, the Manipulation module allows for multi-arm coordination at the highest level with manipulator motion planning and obstacle avoidance at an increasing amount of detail at the lower levels. The broad connectivity (bus structure) within the Mobility, Perception and Manipulation modules allows for at least two levels of representation, i.e., hard-wired or reflex response (e.g., the global route planner can directly control the wheel motion without passing through the Local Navigation and Obstacle Avoidance routines) and problem solving/classification higher level symbolic reasoning (e.g., scene interpretation). In fact, this two level description is arbitrary and corresponds to a continuum of representations for problem solving.

Within their own domain, the Mobility, Perception, and Manipulation modules proceed asynchronously reading from and writing to a Public Knowledge repository when appropriate. Recommended commands to the robot primitives are sequenced temporally and monitored by the Automated Monitor within the overall control module.

The architecture proposed in Fig. 3 is intended to provide capability toward performance which would customarily be deemed "intelligent". It was not conceived as an attempt to parallel the operation of the human brain; thus, the research envisioned strives more toward autonomous robotics and artificial intelligence rather than cognitive science.

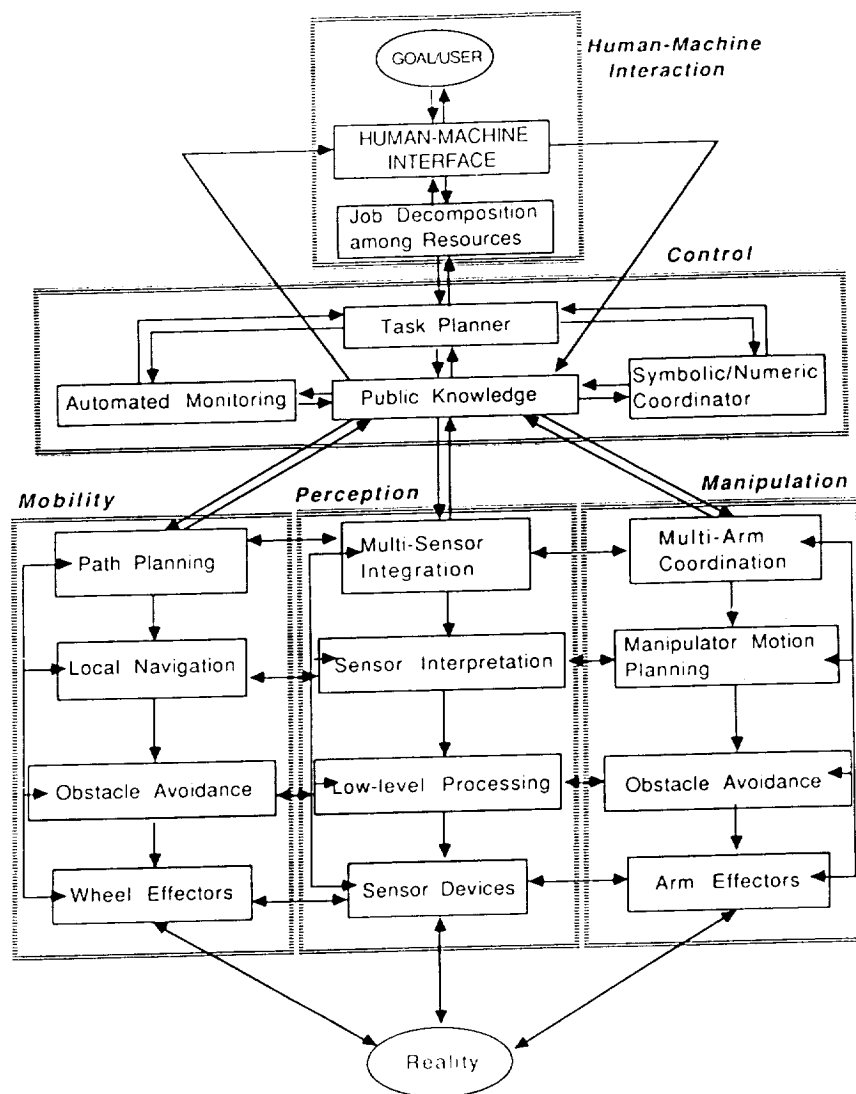


Fig. 3. An initial characterization of the HERMIES-III Software Architecture.

The architecture enables us to investigate a number of fundamental issues including the relative exploitation of algorithms vs. heuristics, the degree of generalized problem solving and learning vs. the specialized knowledge intensive domain approach, the degree of high-level reasoning vs. “wired” reflex, and the issues of long-term vs. short-term and decentralized vs. centralized memory. We anticipate that the HERMIES-III robot studies will lead to contributions to the understanding of these important issues.

### C. Experiments with HERMIES-III

A paradigm problem has been chosen involving the operation, replacement and repair of valves, such as are encountered in an industrial environment. These tasks were deemed generic in nature, involving capabilities directly applicable to many other tasks. Some examples follow.

- Radiation monitoring – Mobile robot moves a monitoring device across surfaces to be surveyed.
- Decontamination – Brushing or spraying objects of various sizes and shapes.
- Erecting shielding – Stacking lead bricks.
- Changing an inserted module (filter, etc.) – Disassembly and assembly of dissimilar component parts. Many such tasks require two arms.
- Operation, replacement and repair of valves – Important tasks in industrial environments.

HERMIES-III is (a) instructed to navigate in an unstructured (a priori unspecified) environment, (b) find a control panel, (c) diagnose the problem(s) by reading meters and observing status of buttons that can be illuminated, (d) navigate through a complex piping network to location of valve(s), (e) adjust position of valve(s) to alleviate the problem, (f) return to the control panel to verify that the problem has been solved, and (g) return to the initial location.

An example of such an environment is illustrated in Fig. 4 in which HERMIES-III is shown performing manipulations.

Experimental features include the following:

- Multiple valves, pipes, meters; control panel; obstacles on floor;
- Exact world model (as-built drawings) not given, only information equivalent to preliminary drawings;
- Valves occluded from view and obstructed;
- Current operability of valves unknown;
- Multiple navigation paths of differing complexity;

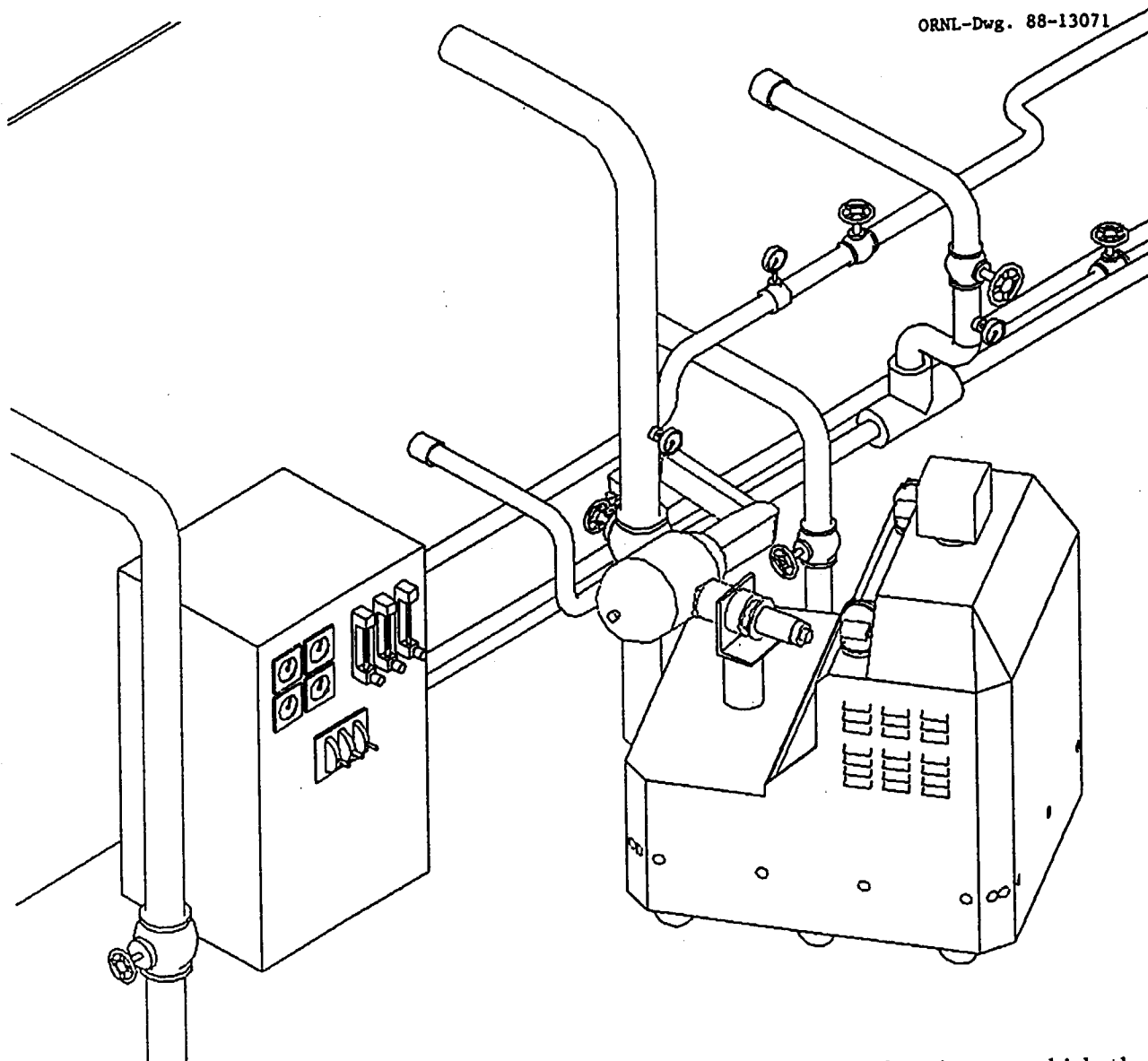


Fig. 4. HERMIES-III operates a valve, access to which is obstructed both by the pipe on which the valve is mounted and another pipe to the left. Visual location of the valve must be provided either by use of the body-mounted cameras from another position of the robot, or by a wrist-mounted camera (not shown).

- Completion of some tasks subject to time constraints.

The environment (network of pipes, etc.) described above will have a number of variable parameters so that not one but a whole range of experiments may be done. The equipment will be constructed so that its parameters can easily be varied. It will be possible to vary the orientations of the valves so that they point up, down, horizontally, or at any other angle, and so that their axes are not necessarily perfectly perpendicular to those of the pipes to which they are attached. Where a valve is obstructed or occluded by another object such as a pipe, the relative positions and orientations of the valve and the obstructing object will be variable. If a valve is behind an access window, its position and orientation relative to that window will be variable.

The intent of these experiments is to highlight research achievements including, (1) multiple cooperative autonomous robots, (2) multi-tasking including smooth continuous motion and simultaneous sensor data processing, (3) ability to deal with real-time asynchronous unexpected events within the framework of a parallel expert system, (4) multi-sensor integration for 3-D navigation and manipulation, and (5) human scale manipulation using CESARm.

### III. SAFETY

Because of the size and complexity of HERMIES-III, safety considerations have played an important role in its design. These fall into several categories as follows.

#### A. Mechanical Safety

The CESAR Laboratory has restricted access in the operating region of HERMIES-III. On-board strobe lights indicate the availability of motor power, i.e., that HERMIES-III is capable of motion. Possible causes for collision accidents involving the chassis or manipulator include errors in computer programs and malfunctions of the computer or motor-drive hardware. Safety measures include the following, some of which are implemented through hardware interlocks for the main motor power.

- A key-switch interlock prevents all but authorized experimenters from operating the vehicle.
- Operation of the robot is conducted with a minimum of two persons, one of whom has the sole duty of keeping depressed a radio-linked "dead man's" switch while closely observing all robot motion. Loss of radio contact terminates motor power.
- There is a large, red, easily accessible kill switch on each of the vehicle's four corners.
- A computer-actuated signal is used to cause a power-down of all motors in the event of a computer crash, which could lead to a loss of control.
- Computer programs involved in motion are exhaustively checked and tested, and control parameters (e.g., velocities) are tested to ensure that they are within acceptable bounds.



- The sonar transceivers mounted around the entire periphery of the vehicle's base sense objects at distances greater than 3 feet. HERMIES-III is programmed to avoid collision with any sensed object.
- CESARm's joints have brakes which are automatically actuated when the joints are not in motion.

## B. Laser Range Finder

The Odetics Laser Rangefinder (LRF), which operates at 820 nm in the invisible infrared, is interlocked so that the laser is inactivated unless the scanner mirrors are operating. The LRF is then eye safe (Class I) at distances greater than 0.5m. A flashing red light above the LRF alerts experimenters, and restriction of the work area prevents non-experimenters from approaching.

## C. Battery Charging

The lead-acid batteries on-board the vehicle, (equivalent to about 20 automobile batteries) are periodically charged. Hydrogen evolved during charging is removed to the outside through hoses connected to the battery compartments, which are equipped with fans. Air-flow switches are used as interlocks for the battery chargers.

*In summary, the safety considerations involved with a vehicle of this complexity are a vital part of conducting our research in autonomous mobility, manipulation and perception. Our experiments will be carefully phased to assure the required level of reliability and safety.*

## IV. CONCLUSIONS

HERMIES-III is an important testbed for research in autonomous mobility, manipulation and perception. It is comprised of a seven degree-of-freedom manipulator designed for human scale tasks, a laser range finder, a sonar array, an omni-directional wheel driven chassis, multiple cameras, and a dual computer system containing a 16-node hypercube (expandable to 128 nodes) and Motorola 68020 processors. On-board batteries allow for 3-4 hours normal operation. A software architecture which serves as HERMIES-III's "brain" is described with emphasis upon modularity, standardization, and hierarchy.

The current experimental program involves performance of human-scale tasks (e.g., valve manipulation, use of tools), integration of a dexterous manipulator and platform motion in geometrically complex environments, and effective use of multiple cooperating robots (HERMIES-IIB and HERMIES-III). The environment in which the robots operate has been designed to include multiple valves, pipes, meters, obstacles on the floor, valves occluded from view, and multiple paths of differing navigation complexity. The equipment includes a number of variable parameters (e.g., valve orientation, position) so that an entire range of experiments can be accommodated.

The ongoing research program highlights (1) multiple cooperating autonomous robots, (2) multi-tasking including smooth continuous motion and simultaneous sensor data processing, (3) ability to deal with real-time asynchronous unexpected events within the

framework of a parallel expert system, (4) multi-sensor integration for 3-D navigation and manipulation, and (5) human-scale manipulation using CESARm.

## ACKNOWLEDGEMENTS

As a central focus of the CESAR research program, the HERMIES-III development has benefitted from significant review and feedback from the entire CESAR team, the CESAR Advisory Committee, and various independent review groups. Special acknowledgement is due to W. R. Hamel for design conceptualization; R. C. Mann and F. G. Pin for research compatibility and relevance to related programs; S. M. Killough for aid in implementation; P. Spelt and G. de Saussure for review and comment on this paper; and to R. D. Lawson who expertly prepared this manuscript and revised it through several drafts.

## REFERENCES

1. C. R. Weisbin, "Intelligent-Machine Research at CESAR," *AI Magazine*, Spring 1987, Vol. 8, No. 1 (1987). CESAR-87/16.
2. C. R. Weisbin, G. de Saussure and D. W. Kammer, "Self-Controlled: A Real-Time Expert System for an Autonomous Mobile Robot," *Computers in Mechanical Engineering*, Vols. 2, 5, pp. 12-19 (September 1986). CESAR-86/25.
3. W. R. Hamel, S. M. Babcock, M. C. G. Hall, C. C. Jorgensen, S. M. Killough and C. R. Weisbin, "Autonomous Robots for Hazardous and Unstructured Environments," Proceedings of the Robots-10 Conference, Chicago, IL, pp. 5-9 through 5-29 (April 20-24, 1986).
4. B. L. Burks, G. de Saussure, C. R. Weisbin, J. P. Jones, and W. R. Hamel, "Autonomous Navigation, Exploration and Recognition," Winter 1987 Issue of *IEEE Expert*, pp. 18-27. CESAR-87/25.
5. C. R. Weisbin, G. de Saussure, J. R. Einstein, E. Heer and F. G. Pin, "CESAR Research in Autonomous Mobile Navigation and Learning," invited paper for *IEEE Computer*, December 1988. CESAR-88/59.
6. B. L. Burks and P. F. Spelt, "Ongoing Research Using HERMIES - The Hostile Environment Robotic Machine Intelligence Experiment Series," DOE/ANL Training Course on The Potential Safety Impact of New and Emerging Technologies on the Operation of DOE Nuclear Facilities, August 29-September 1, 1988, Idaho Falls, ID. CESAR-88/54.
7. R. V. Dubey, J. A. Euler and S. M. Babcock, "Real-Time Implementation of a Kinematic Gradient Projection Optimization Scheme for Seven-Degree-of-Freedom Redundant Robots with Spherical Wrists," submitted to *Journal of Robotics and Automation*, June 1988. CESAR-88/36.

8. R. V. Dubey, J. A. Euler, and S. M. Babcock, "An Efficient Gradient Projection Optimization Scheme for a Seven-Degree-of-Freedom Redundant Robot with Spherical Wrist," 1988 IEEE International Conference on Robots and Automation, April 25-29, 1988, Philadelphia, Pennsylvania, Proceedings, Vol. 1, pp 28-36. CESAR-87/42.
9. G. N. Saridis, "Toward the Realization of Intelligent Controls," Proceedings of the IEEE, 67, 1115, 1979.
10. J. Albus: "A Control System Architecture for the Space Station Flight Telerobotic Servicer," Proceedings of the Space Telerobotics Workshop, Jet Propulsion Laboratory, Pasadena, California, January 1987.



# First Results in Terrain Mapping for a Roving Planetary Explorer

E. Krotkov, C. Caillas, M. Hebert, I. S. Kweon, T. Kanade

The Robotics Institute  
Carnegie Mellon University  
Pittsburgh, PA 15213

## Abstract<sup>1</sup>

To perform planetary exploration without human supervision, a complete autonomous rover must be able to model its environment while exploring its surroundings. We present a new algorithm to construct a geometric terrain representation from a single range image. The form of the representation is an elevation map that includes uncertainty, unknown areas, and local features. By virtue of working in spherical-polar space, the algorithm is independent of the desired map resolution and the orientation of the sensor, unlike other algorithms that work in Cartesian space. We also describe new methods to evaluate regions of the constructed elevation maps to support legged locomotion over rough terrain.

## 1 Introduction

We are prototyping a legged vehicle called the Ambler (fig. 1) for an exploratory mission on another planet, conceivably Mars, where it is to traverse uncharted areas and collect material samples. Planetary exploration poses significant challenges for rovers: unprecedented levels of autonomy and reliability due to communication delays that limit conventional Earth-based teleoperation; and traversal of rugged, irregular terrain for which existing mechanisms and perception techniques are inadequate.

Papers that describe the background of our work include a comprehensive account of the Ambler configuration [1] and an overview of the integrated research program [4]. The aim of this paper is to describe first results from the Ambler perception system.

The Ambler perception system must build and maintain representations of the terrain and discrete objects—*terrain maps* that are appropriate for a wide variety of tasks, each with different requirements. For example, locomotion and sampling require detailed, local representations, while navigation and mission planning demand broad, global descriptions. In this paper, we do not address the full scope of the perception system; we focus only on building maps based on the observations of a single sensor, and using those maps to support locomotion.

This paper addresses sensing in section 2, and presents a new technique for constructing elevation maps in section 3. It describes methods for analyzing map geometry for locomotion

---

<sup>1</sup>This research was sponsored by NASA under Contract NAGW-1175. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the funding agency.

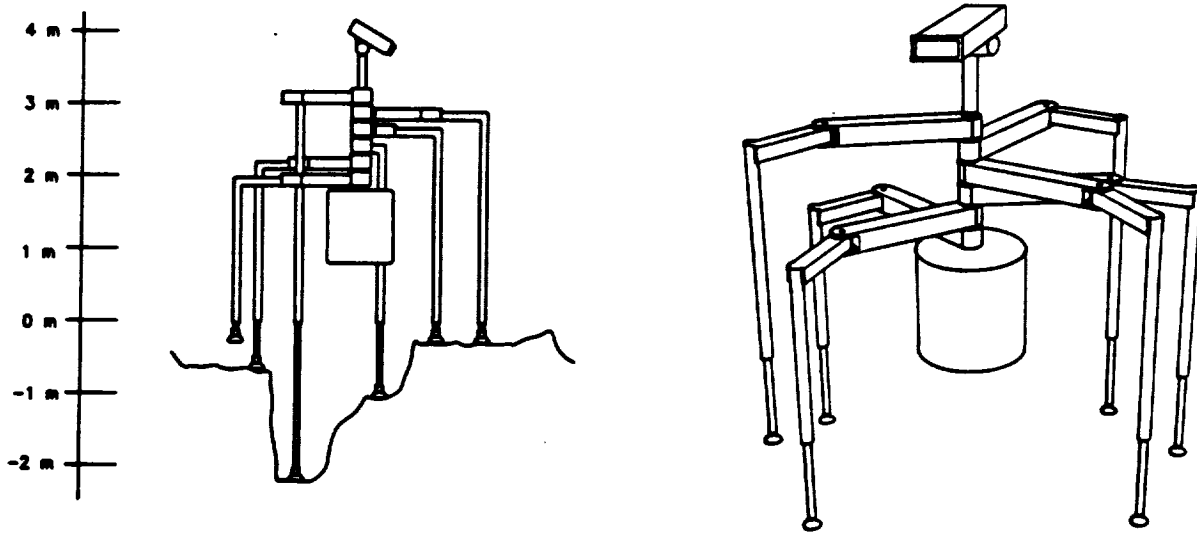


Figure 1: The Ambler drawn at shown scale (left) and another scale (right)

in section 4, and documents experimental methods and results in section 5. It concludes by discussing limitations and future work.

## 2 Active Range Sensing

The Ambler perception system will use multiple sensing modalities, both imaging and non-imaging. Here we concentrate on active range sensors, which measure the distance to an object in the environment by observing the reflection of a reference signal (sonar, laser, radar, etc.) from the object. Active sensors offer two chief advantages: they provide range data without the numerous computations required by passive techniques such as stereo vision; and they are largely insensitive to illumination conditions, thus simplifying the image analysis problem.<sup>2</sup>

We use a scanning laser range finder, developed by ERIM, that measures the phase difference between an amplitude-modulated laser beam and its reflection from a point in the scene [7]. We measure the coordinates of the point in a non standard spherical polar reference frame, in which  $\rho$  is the measured range, and  $\phi$  and  $\theta$  are the vertical and horizontal scanning angles of the beam direction corresponding to row and column position in the image. The Cartesian coordinates of a point measured in spherical polar coordinates have been derived [3] as

$$x = \rho \sin \theta \quad , \quad y = \rho \cos \phi \cos \theta \quad , \quad z = \rho \sin \phi \cos \theta \quad . \quad (1)$$

<sup>2</sup>This is especially important for images of outdoor scenes in which illumination can be neither controlled nor predicted.

### 3 Elevation Map Construction

Applying eq. 1 to the measurements in a range image yields an elevation map. However, this map is non-uniform in Cartesian space, because the coordinate transformation is non-linear. Further, the map grows less dense and less accurate with increasing distance from the sensor.

One could circumvent the former difficulty by using a map structure that is not a regularly spaced grid, such as a Delaunay triangulation [5]. However, this is not practical because of the complex algorithms required to access data points and their neighborhoods.

Another approach is to interpolate between data points to build a dense elevation map on a grid, either by approximating the surface between data points (e.g., as a bicubic surface), or by globally fitting a surface under some smoothness assumptions (e.g., regularization). However, both of these approaches have significant limitations: they make assumptions on the local shape of the terrain which may not be valid in the case of rough terrain; and they depend heavily on the resolution and position of the grid (i.e., they cannot compute an estimate of the elevation at an  $(x, y)$  position that is not a grid point without resampling the grid).

We propose an alternative, the locus algorithm, that uses a model of the sensor to interpolate at *arbitrary resolution* without making any assumptions on the terrain shape other than the continuity of the surface.

#### 3.1 Locus Algorithm

The problem of finding the elevation  $z$  of a point  $(x, y)$  is equivalent to computing the intersection of the surface observed by the sensor with the vertical line passing through  $(x, y)$ . The basic idea of the locus algorithm is to convert the latter formulation into a problem in image space<sup>3</sup> (fig. 2). A vertical line<sup>4</sup> is a locus (curve) in image space, whose equation as a function of  $\phi$  is derived by inverting eq. 1, assuming  $x$  and  $y$  constant:

$$\rho = \rho_I(\phi) = \sqrt{\frac{y^2}{\cos^2 \phi} + x^2} \quad , \quad \theta = \theta_I(\phi) = \arctan \frac{x \cos \phi}{y} \quad . \quad (2)$$

Similarly, the range image can be viewed as a surface  $\rho = I(\phi, \theta)$  in  $\phi, \theta, \rho$  space. The problem then is to find the intersection, if it exists, between a curve parameterized by  $\phi$  and a discrete surface. Since the surface is known only from a sample of data, the intersection cannot be computed analytically.

Instead, we must search along the curve for the intersection point. Let  $\hat{\theta}_I(\phi)$  be the image column closest to  $\theta_I(\phi)$ , and let  $\Delta(\phi_j) \equiv \rho_I(\phi_j) - I(\phi_j, \hat{\theta}_I(\phi_j))$ . The search proceeds in two stages. First, we locate the two scanlines of the range image,  $\phi_1$  and  $\phi_2$ , between which the intersection must be located, i.e., such that  $\text{sgn} \Delta(\phi_1) \neq \text{sgn} \Delta(\phi_2)$ . Second, we apply a binary search between  $\phi_1$  and  $\phi_2$ . The search stops when  $|\phi_n - \phi_{n+1}| < \epsilon$  (i.e., the resolution of the elevation is controlled by the parameter  $\epsilon$ ). Third, since there are no pixels between  $\phi_1$  and  $\phi_2$ ,

<sup>3</sup>Specifically, spherical-polar space rather than row-column space.

<sup>4</sup>We have generalized the locus algorithm from the case of a vertical line to the case of a general line in space [3], which allows us to build maps using any reference plane, not just the  $xy$  plane. We present the case of the vertical line to simplify exposition.

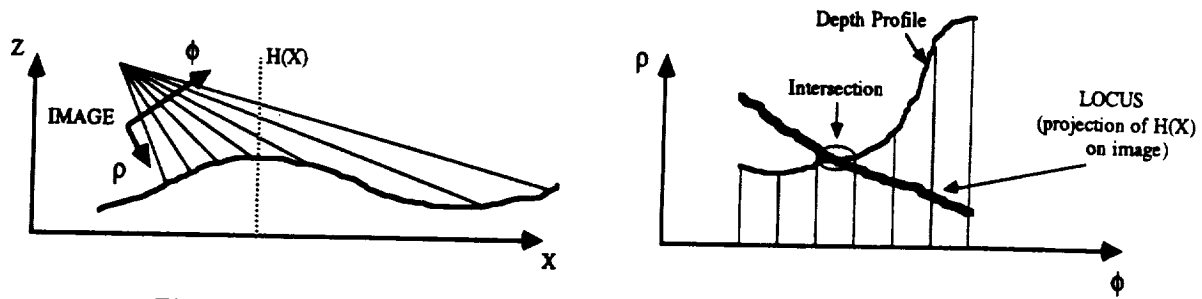


Figure 2: Imaging geometry (left) and one-dimensional locus (right)

we perform Lagrangian interpolation for  $\phi_1 < \phi < \phi_2$ , using as control points the four pixels that surround the intersection point.

The result is a value  $\phi$  that is mapped to  $\rho$  and  $\theta$  by eq. 2, and then mapped to an elevation value by eq. 1. Repeating this for vertical lines at every desired  $(x, y)$  point yields a dense elevation map of the desired resolution, as required.

### 3.2 Range Shadows

Objects in the environment may cast range shadows (cause occlusions). It is important to identify the occluded regions, because if we apply the locus algorithm there directly, then the surface would be smoothly interpolated, possibly incorrectly. In turn, this could lead the rover to plan a path through that region, expecting it to be traversable when in fact it is unknown.

One could detect empty regions in the elevation map given by eq. 1, without interpolation. This does not work, because the size of the shadow regions may be on the order of the average distance between data points.<sup>5</sup>

Another approach is to incorporate the detection of shadow regions into the locus algorithm, again working in image space. We observe that a range shadow corresponds to an occluding edge in the image. As in fig. 3, an  $(x, y)$  location in the map is in a shadow area if its locus intersects the image at a pixel that lies on such an edge. We implement this idea by first detecting edges in the range image by using the GNC algorithm [2]. Then, when we apply the locus algorithm and observe that the locus of a given location intersects the image at an edge pixel, we mark that location as lying in a range shadow.

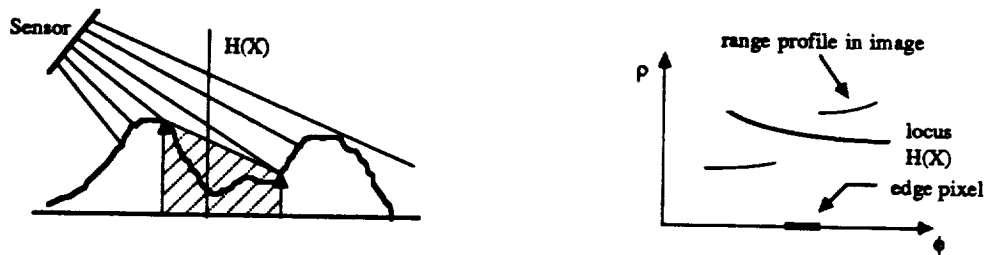


Figure 3: Shadowed area (left) and range discontinuity (right)

<sup>5</sup>This is especially true for distant regions in which the distribution of data points is sparse.



### 3.3 Uncertainty

We have developed a probabilistic model of the uncertainty on the sensor measurements, according to which the measured range errors are normally distributed with standard deviation proportional to the square of measured range ([3], p. 7). The range measurement uncertainty is oriented along the direction of measurement (fig. 4).

To identify the uncertainty on the elevation value at each grid point  $(x, y)$ , as part of the locus algorithm we transform the uncertainty on a sensor measurement so that it is oriented along the  $z$  axis ([3], pp. 25-27). This conversion is non trivial, since the the range uncertainty is distributed across a region in the elevation map. According to this model, the distribution of elevation errors is approximately normal, with standard deviation proportional to the product of measured range and elevation.

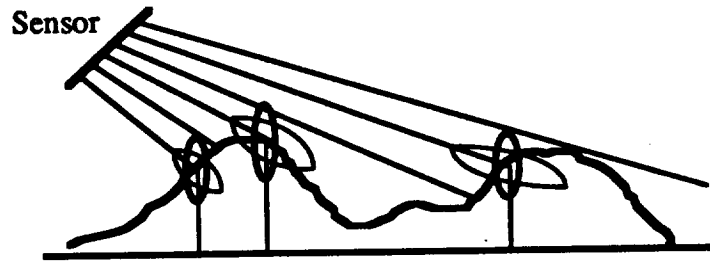


Figure 4: One-dimensional uncertainty distributions on sensor and map

## 4 Footfall Evaluation

A perceptual task unique to legged locomotion is to evaluate terrain regions as footfall locations (foot placements). This is essential for locomotion over the rugged terrain that could be encountered on the surface of other planets such as Mars. In this section, we describe several methods to evaluate elevation map regions as footfall locations. These methods operate on the geometric structure of the surface described by the elevation map, for now ignoring important material properties of the soil such as load-bearing strength, compliance, and coefficient of friction. While incomplete, these methods are considerably more sophisticated than others reported in the literature, which require operator interaction [6].

An Ambler foot is modeled by a flat disk 30 cm in diameter. The problem is to find the “best” foot-shaped subregion  $B$  in a given region  $R$  of a given elevation map.<sup>6</sup> We have developed five solutions, corresponding to different measures of “best,” and present them in increasing order of sophistication.

**Max-min** Find  $B$  that minimizes the difference  $z_{max} - z_{min}$  of extremal elevations, as illustrated in fig. 5a. There are cases where this method prefers a flat surface punctuated by a single spike rather than an undulating surface (cf. figs. 5a and 5b). This is obviously undesirable.

---

<sup>6</sup>The region  $R$  is computed elsewhere based on the current heading and gait.

**Planar fit** Find  $B$  that best fits a plane, subject to the constraint that the plane normal is approximately parallel to the leg. This method suffers the same deficiency as above.

**Support area** Find  $B$  that minimizes the depth of penetration  $d_{opt}$  into the soil (fig. 5c) required to achieve the minimum necessary support area  $A_{min}$  (contact area between foot and terrain, or the number of map points within the circumference of the disk that are above the plane of the foot). This method is superior to the previous two to the extent that it better accounts for the shape of the terrain. However, there are cases that it fails to distinguish, e.g., two sinusoidal surfaces with the same frequency but different amplitudes. This method should, but does not, select the surface with smaller amplitude variations.

**Free volume** Find  $B$  that minimizes the free (unoccupied) volume between the foot and soil,  $V \equiv Nz_{max} - \sum_{i=1}^N z_i$ , as shown in fig. 5b. This method correctly discriminates the two sinusoidal surfaces described above. However, it does not take into account the distribution of “holes” in the surface or the consequences of applying force to (stepping on) the surface.

**Equilibrium** Find  $B$  that minimizes  $V$  and  $E$ , where  $E \equiv \sqrt{m_x^2 + m_y^2}$  is the first moment of the mass distribution about the center of the foot, and  $m_x = \sum_{i=1}^N x_i(z_{max} - z_i)$ . The second condition ensures the footfall of greatest “equilibrium” (balance) with respect to holes in the surface, as suggested by fig. 5d. The idea is that as the foot contacts sandy soil, the sand fills the holes with a minimum of foot penetration into the soil, and as the foot contacts rocky soil, it exerts the minimum lateral forces on potentially unstable materials.

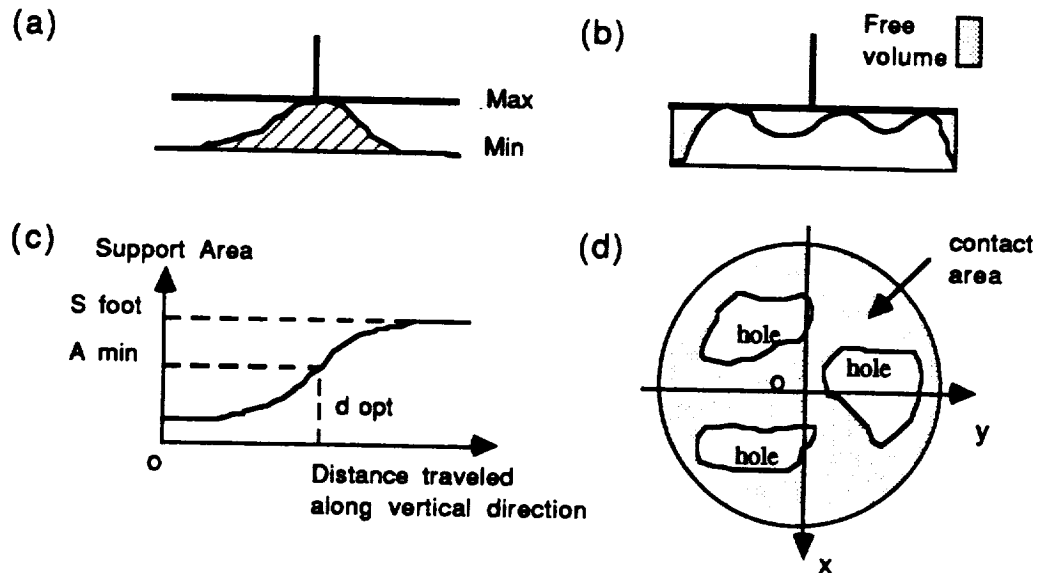


Figure 5: Footfall evaluation criteria

## 5 Experiments

In this section we summarize our initial experiments and results.

First, we evaluated the locus algorithm on synthesized range images with additive Gaussian noise by comparing its performance to that of Cartesian space interpolation algorithms (cf. section 3). The results show that the locus algorithm is more stable with respect to surface orientation and noise level than the others ([3], p. 25). We conclude that this is due to performing the interpolation in image space instead of first applying eq. 1 to the data points.

Then, we tested the locus algorithm on a variety of real range images. The left half of fig. 6 shows the result of applying it to a range image of uneven terrain found at a construction site. The figure shows the original range image and displays the elevation map as an isoplot surface at 10 cm resolution. The right half of fig. 6 shows an overhead view of a different elevation map, where the grey levels indicate the following: white is shadow, black is unknown, grey is proportional to elevation uncertainty. Note that more distant points are more uncertain, as expected.

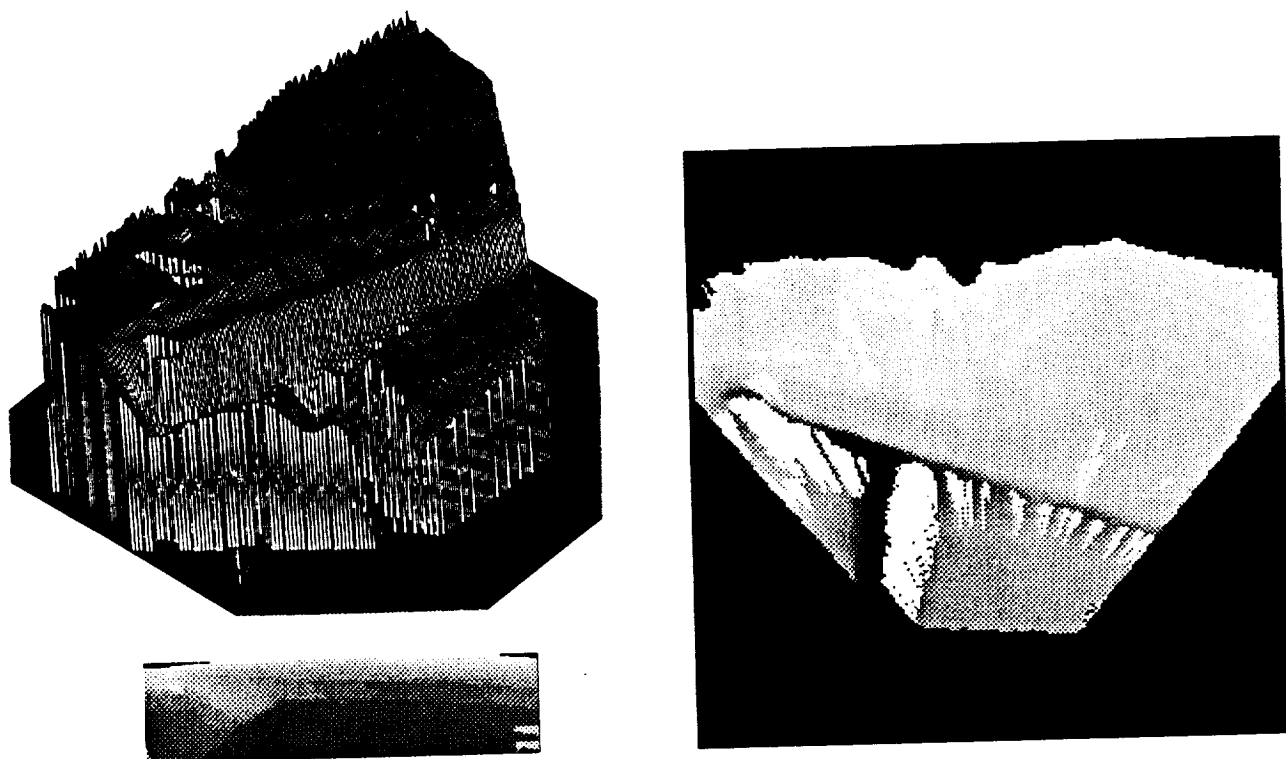


Figure 6: Elevation map (left), shadow regions and uncertainty (right)

Finally, we tested a partially integrated Ambler system at an experimental testbed (fig. 7): a single leg with a fully operational controller; the range finder mounted above the leg; and a 25 m<sup>2</sup> "sandbox" of terrain to be traversed. The perception system communicates with other modules through queries, which typically are requests for the elevation map at a given resolution within a polygonal region.

The lower left panel of fig. 8 shows the polygonal region referenced by queries for elevation, uncertainty, and footfall location, and the other panels show the perception system's replies

ORIGINAL PAGE  
BLACK AND WHITE PHOTOGRAPH

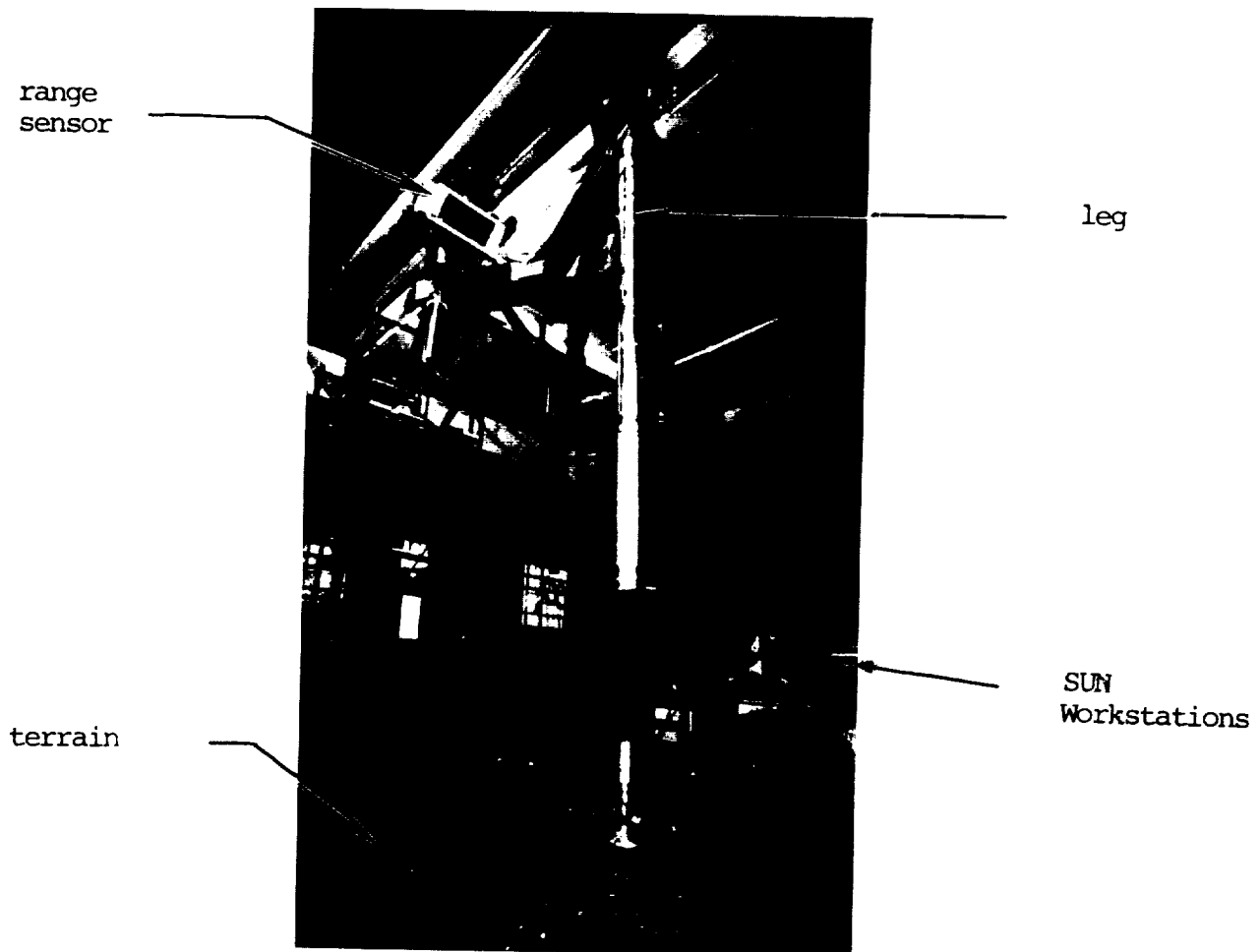


Figure 7: Single leg testbed

ORIGINAL PAGE IS  
OF POOR QUALITY

computed by the locus and equilibrium algorithms. We evaluate the selected footfall location by servoing the leg there, thus closing the loop between perception and action. Visual inspection of the servoed positions shows the selected locations to be reasonably accurate; quantitative error measurements are not yet available. Dozens of trials on different terrains suggest that the perception algorithms provide reliable and reasonably accurate descriptions of the terrain that suffice for moving the leg and executing footfalls.

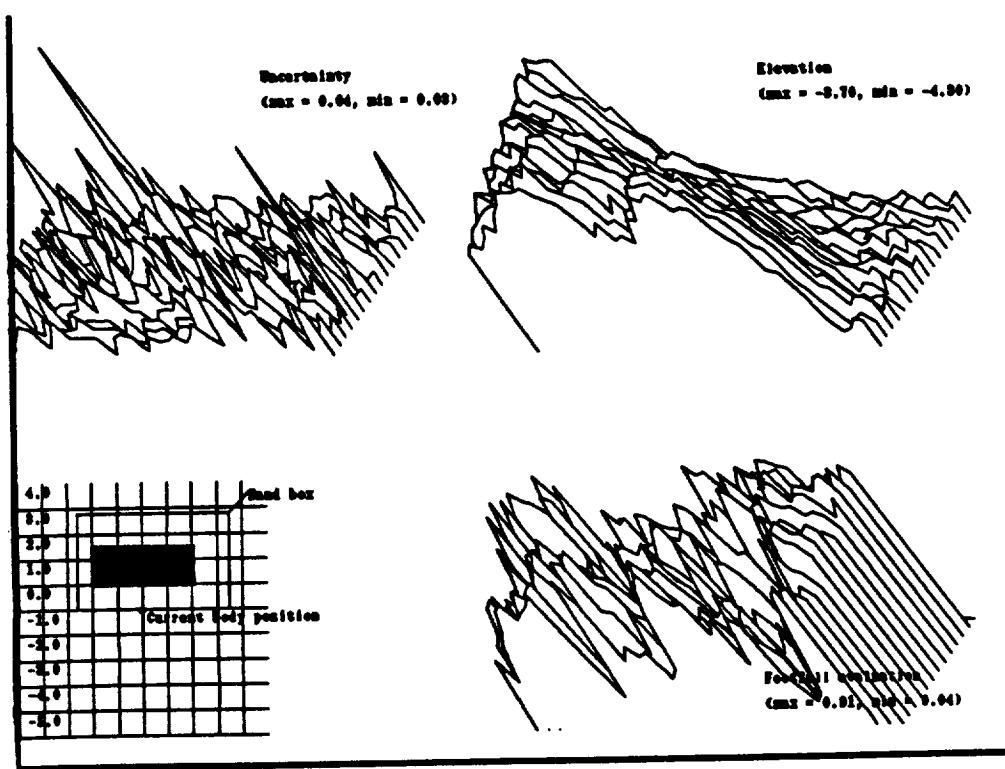


Figure 8: Perception system replies to map and footfall location queries

## 6 Discussion

In this paper we presented techniques to build maps based on the observations of a single range sensor, and to use those maps to support locomotion: a new algorithm to build elevation maps at arbitrary resolution, including elevation uncertainty and unknown areas; and new methods for geometrically evaluating areas of the constructed elevation map as footfall locations.

Preliminary experiments demonstrate that an integrated system can build and use maps to select footfall locations. This illustrates the advantages of working in image space rather than in Cartesian space.

While the first results are encouraging, further work is required both in map building and map analysis. For the former, we must complete an automatic calibration procedure to more

accurately relate sensor and vehicle coordinate systems. For the latter, we must investigate more sophisticated footfall evaluations that take into account not only the local geometry of the terrain, but also geometric uncertainty and material properties of the soil such as load-bearing strength, compliance, and coefficient of friction. Further, we must better integrate the algorithms into the Ambler system, and make more quantitative assessments of their performance.

The work reported in this paper addresses a small fraction of the problems faced in developing a complete perception system for the Ambler. The scope of future research includes two broad categories: navigation and sampling. For the former, we aim to increase map coverage by processing multiple views from multiple sensors, to determine vehicle position by landmark triangulation, and to compute vehicle displacement by matching elevation maps. For the latter, we intend to use surface topography to identify promising sample sites, and to build models of discrete objects both to select particular samples and to guide sample acquisition.

## References

- [1] J. Bares and W. Whittaker. Configuration of an Autonomous Robot for Mars Exploration. In *Proc. World Robotics Conference*, Society of Mechanical Engineers, To appear, May 1989.
- [2] A. Blake and A. Zisserman. *Visual Reconstruction*. MIT Press, Cambridge, Massachusetts, 1987.
- [3] M. Hebert, T. Kanade, and I. Kweon. *3-D Vision Techniques for Autonomous Vehicles*. Technical Report CMU-RI-TR-88-12, The Robotics Institute, Carnegie Mellon University, 1988.
- [4] E. Krotkov, J. Bares, M. Hebert, T. Kanade, T. Mitchell, R. Simmons, and W. Whittaker. An Autonomous Rover for Exploring Mars. *IEEE Computer*, To appear, June 1989.
- [5] D. J. Orser and M. Roche. The Extraction of Topographic Features in Support of Autonomous Underwater Vehicle Navigation. In *Proc. Fifth International Symposium on Unmanned Untethered Submersible Technology*, Merrimack, New Hampshire, June 1987.
- [6] F. Ozguner, S. J. Tsai, and R. B. McGhee. An Approach to the Use of Terrain-Preview Information in Rough-Terrain Locomotion by a Hexapod Walking Machine. *International Journal of Robotics Research*, 3(2):134-146, Summer 1984.
- [7] D. Zuk, F. Pont, R. Franklin, and V. Larrowe. *A System for Autonomous Land Navigation*. Technical Report IR-85-540, Environmental Research Institute of Michigan, Ann Arbor, Michigan, 1985.

## PLANETARY ROVER TECHNOLOGY DEVELOPMENT REQUIREMENTS

Roger J. Bedard Jr. and Brian K. Muirhead  
 Jet Propulsion Laboratory  
 California Institute of Technology  
 4800 Oak Grove Drive  
 Pasadena, California 91109

and  
 Dr. Melvin D. Montemerlo and Murray S. Hirschbein  
 National Aeronautics and Space Administration  
 Office of Aeronautics and Space Technology  
 Washington D.C. 20546

### ABSTRACT

Planetary surface (including lunar) mobility and sampling capability is required to support proposed future National Aeronautics and Space Administration (NASA) solar system exploration missions.

The NASA Office of Aeronautics and Space Technology (OAST) is addressing some of these technology needs in its base research and development program, the Civil Space Technology Initiative (CSTI) and a new technology initiative entitled "Pathfinder". The Pathfinder Planetary Rover (PPR) and Sample Acquisition, Analysis and Preservation (SAAP) programs will develop and validate the technologies needed to enable both robotic and piloted rovers on various planetary surfaces.

This paper discusses the technology requirements for a planetary roving vehicle and the development plans of the PPR and SAAP programs.

### 1. INTRODUCTION

A planetary surface (including lunar) mobility and sampling capability is required to support proposed future National Aeronautics and Space Administration (NASA) solar system exploration missions.

The Mars Rover Sample Return (MRSR) project is the earliest NASA project identified as needing planetary rover mobility and sample return technology. MRSR is currently targeted for a late 1990's launch. The value of planetary landers in surveying future landing sites for manned missions was demonstrated by the Surveyor lunar missions. The value of planetary landers in performing scientific exploration was demonstrated by the Surveyor and the Viking Mars mission. Surveyor and Viking collected scientific data within local areas about the landing sites. In many cases, however, areas of scientific interest are likely to be in regions remote from acceptable safe spacecraft landing sites.

Manned and unmanned rover technology to support exploration, mining and construction are required for the crewed Lunar and Mars missions.

The NASA Office of Aeronautics and Space Technology (OAST) is addressing some of these technology needs in its base research and development program, the Civil Space Technology Initiative (CSTI) and a new technology initiative entitled "Pathfinder". The Pathfinder Planetary Rover (PPR) and Sample Acquisition, Analysis and Preservation (SAAP) programs will develop and validate the technologies needed to enable scientific and exploration missions by robotic and piloted rovers on various planetary surfaces. In its first phase, the program will be focused on automated, unmanned rover technologies needed for a Mars rover sample return type mission.

The eventual need for autonomous navigation, autonomous sample acquisition, robust mobility, low mass electrical power, fault tolerant computing, high bandwidth communications and mission operations autonomy are enabling technologies which affect the vehicle's travel range and science return. The implementation of these requirements forces advances in several areas of technology; namely:

- o Mobility,
- o Navigation,
- o Sample Acquisition, Analysis and Preservation,
- o Mission Operations,
- o Computation,
- o Power,
- o Temperature Control,
- o Communication.

Development and integration of these technologies will allow orders of magnitude increase in the effectiveness of remote surface operations.

## **2. MOBILITY**

A mobility system must combine locomotion, stability and ruggedness over a wide variety of terrains with acceptable power consumption and reasonable control requirements. A number of experimental locomotion concepts which appear suitable for planetary surface operations have been built and tested. These concepts encompass wheeled, legged and hybrid configurations.

The state-of-the-art is:

- o Wheeled locomotion with moderate mobility characteristics,
- o Three dimensional (3-D) vehicle / terrain modeling for higher speeds (commercial and military vehicles) than those of interest,
- o Lunar rover wheel packaging and deployment concepts.

The mobility technology needs include:

- o A practical, high mobility locomotion system with low mass and power consumption and automated control capability (reasonable control requirements),
- o A general modeling capability for vehicle / terrain obstacle climbing is needed for comparative assessment of locomotion options, vehicle dynamics and control verification and expectation generation/execution monitoring,
- o Deployable locomotion structure concepts / technologies that offer efficient packaging of the vehicle within the volume constraints of the aeroshell,



- o Adaptable locomotion structure concepts / technologies that allow flexibility in response to different terrain and rover operational states.

### 3. NAVIGATION

Because of the long signal time to Mars (6-44 minutes round trip), it is impractical to continuously teleoperate a Martian Rover from Earth (one on which individual movements are controlled from Earth). Therefore, some navigation autonomy on the Rover is needed. A highly autonomous rover capable of traveling safely over long distances for many days in unfamiliar terrain without guidance from Earth is well beyond the present state-of-the-art. In between the extremes of continuous teleoperation and highly autonomous, various degrees of autonomy are possible. Two in particular; namely, computer aided remote driving (CARD) and semiautonomous navigation (SAN) have been identified as feasible with additional technology development.

With CARD, stereo pictures from the rover are sent to Earth where they are viewed by a human operator using a stereo display. The operator designates a safe path for the vehicle to follow as far ahead as can be seen. This plan is sent to the rover which executes the path by dead reckoning navigation aided by computer vision. A new stereo pair of pictures is taken from the new position and the process repeats itself. Depending on the terrain, the rover might travel 5-30 meters on each of these iterations. Assuming 2-7 command cycles from earth per day, the daily traverse would be 10-200 meters.

In the SAN method, local paths are planned autonomously (without interaction from humans on Earth) using images obtained on the vehicle, but they are guided by global routes planned less frequently by humans on Earth. These global routes are developed from a topographic map produced from images obtained by an orbiting satellite.

The sequence of operations, in the portion of SAN involving Earth, is as follows. As commanded from Earth, the orbiter takes a stereo pair of pictures (by taking the two pictures at different points in the orbit) of an area to be traversed. A spatial resolution of about 1 meter is desired. The pictures are sent to Earth where they are used by a human to plan an approximate route for the vehicle to follow designed to avoid large obstacles, dangerous areas and dead-ends. This route and a topographic map for the surrounding area are sent from Earth to the rover. The process repeats, as needed; perhaps once for each traverse between major sites where experiments are to be done, or perhaps once per day or so on long traverses.

The sequence of operations, in the portion of SAN taking place on Mars, is as follows. The rover views the local scene and, by using automatic stereo correlation or laser ranging, computes a local topographic map. This map is matched to the portion of the global map sent from earth for purposes of position determination. The high resolution local map is analyzed by computation on the rover to determine the safe areas over which to drive. A new plan is then computed, revising the approximate route from the Earth. Using the revised path, the rover then drives ahead a short distance (perhaps 5-10 meters), and then the process repeats. With SAN operation, a daily travel of 700-7000 meters is feasible.

The state-of-the-art permits rudimentary demonstration of CARD and SAN technology (no proximity contact sensing, surface property determination, expectation generation, execution monitoring, etc).

The navigation technology needs include:

- o World sensing and perception accomplished via multiple sensor and algorithm fusion weighted by certainty, time and sensor source,
- o Surface property determination via correlation of contact and non-contact sensor data and algorithmic / heuristic driven responses to results,
- o Robust path planning using multiple models and degraded terrain knowledge bases,
- o Expectation generation and execution monitoring and replanning for dynamic response to uncertainty in sensing and perceptual data.

#### 4. MISSION OPERATIONS

The U.S. has operated a roving vehicle on the surface of another planetary body (the moon) with the direct involvement of a human driver (Apollo Program) and has operated an unmanned stationary lander on the surface of Mars (Viking). The U.S.S.R. has operated an unmanned roving vehicle on the surface of the moon (Lunakod). The operation of an autonomous unmanned rover on the surface of planet tens of light minutes away, however, represents an entirely unproven technology. In addition, the operations of the various rover subsystems (mobility, navigation, power, sampling, communication, etc) are highly interdependent and quite complex; thus providing a significant system operation challenge.

In order to maximize mission effectiveness, a high level of both onboard and ground based autonomy is essential.

The state-of-the-art is represented by Galileo uplink command generation technology which is oriented towards repetitive tasks and / or one-time tasks of short duration and Galileo onboard autonomy consisting of a few isolated autonomous low level subsystems.

The ground operations technology needs include an uplink command generation of non-repetitive, long-duration tasks with a couple orders of magnitude improvement in present command cycle turnaround times.

The onboard operations technology needs include the processing of high level goals at the system level.

#### 5. COMPUTATION

JPL experience in past spacecraft projects and the premise for future projects is that mission capabilities are limited by the performance of onboard computers. Thus it is essential to maximize the computer capability for future spacecraft missions.

Unmanned rover computation requirements include general purpose and special purpose processing, fault tolerance, fast processing speed, low power consumption, low mass, space qualified computers organized in a distributed, parallel processing architecture. High speed, high capacity data storage is also required.

The state-of-the-art is represented by the Mariner Mark II (MM II) flight computer (32032 processors with 0.25 million instructions per second (MIPS)/processor, 20 watts/MIPS and 300 components/processor) and digital tape recorder (sequential access) technology.

The computation technology needs include:

- o 5-10 MIPS general purpose computer with 1-4 MIPS/processor, 4-8 watts/MIPs and 50-100 components/processor,
- o 200 MIPS (or equivalent) special purpose image processor,
- o High performance, nonvolatile, random access data storage.

## 6. POWER

A planetary requires a compact, lightweight, very high capacity onboard power system. A Radioisotope Thermal Generator (RTG) is the preferred heat source for the power levels required for an unmanned rover vehicle (ie, about 500 - 1000 watts). Advanced thermoelectric multicouples provide the thermal to electric energy conversion. Advanced sodium sulfide or lithium titanium disulfide batteries supplement the RTGs during periods when increased power is required for higher speeds, obstacles or increased slopes. Reduced mass / volume power conditioning / control integrated circuit elements are also required.

The state-of-the-art is represented by:

- o Galileo RTGs, with a specific power of 5 W/Kg, are designed for operation in the vacuum of space; there is no existing power source / conversion technology capable of operating in an atmosphere (Viking RTGs, although not available today, had a specific power of about 2 W/Kg),
- o Current energy storage technology (nickel hydrogen batteries) provides a specific energy of 45 Whr/kg,
- o Galileo power conditioning / control elements (discrete components) provide a specific power of 12 W/kg and a power density of 0.06 W/cm<sup>3</sup>.

The power technology needs include:

- o A planetary surface RTG energy source and thermal to electric conversion system with a specific power of 10 W/kg,
- o Increase the specific energy of storage components (to 100 Whr/kg),
- o Increase specific power and power density of conditioning / control elements (to 21 W/kg and to 0.5 W/cm<sup>3</sup>).

## 7. TEMPERATURE CONTROL

A planetary rover requires an efficient thermal transfer and control system for maintaining rover elements within temperature limits for all environmental exposures from pre-launch through surface operation. Thermal energy storage may be required during aerocapture or during high power dissipation or adverse environmental conditions.

The state-of-the-art is represented by conventional spacecraft two phase thermal control materials and devices.

The planetary rover thermal control technology needs include the development of two phase heat transfer loop technology for application to planetary surface operation (gravity, atmosphere, dynamics, etc.) and higher efficiency temperature control materials that operate in a hostile planetary atmosphere.

## **8. COMMUNICATIONS**

A Ka-band (32 GHz) communication downlink (from Mars to Earth) offers significant advantages compared to existing X-band (8.4 GHz) technology. These advantages include higher data rate, greater link reliability, smaller antenna size (important for packaging in the aeroshell) and increased availability (X-band communications from Mars would require the 70 meter Deep Space Network (DSN) stations with a forecasted availability of 30% in the late 1990's as compared to the forecasted 100% availability using Ka-band and the 34 meter DSN stations).

The state-of-the-art is:

- o Laboratory demonstrations of small monolithic millimeter arrays,
- o X-band solid state and traveling wave tube (TWT) amplifier,
- o Ka-band TWT efficiency of about 20%,
- o Commercial Ka-band field effect transistors (FET) power of about 0.15 W.

The communication technology needs include:

- o A Ka-band monolithic transmitter phased array with over 100 elements producing 40 W of output power and greater than 30% overall DC to RF conversion efficiency,
- o Millimeter wave integrated circuit (MMIC) digital data phased array signal distribution system,
- o High DC to RF conversion efficiency MMIC (40%) and TWT amplifiers (45%) operating at 32 GHz.

## **9. SAMPLE ACQUISITION, ANALYSIS AND PRESERVATION**

The primary function of an unmanned or piloted rover in any planetary exploration mission is to provide the mobility to enable exploration and to conduct scientific surveys. The technology developed under the SAAP program is intended to be carried by a rover as part of science and exploration missions, and as such, the SAAP and PPR programs must be well coordinated.

The rover must have the ability to identify promising sites which contain scientifically interesting surface samples. It then must have the ability to acquire the desired samples. This will require imaging and ranging instrumentation to provide multi-spectral data for precise sample location and a robotic system to acquire the samples. Once acquired, analytical equipment onboard the rover will determine the sample's elemental, chemical and physical properties in order to determine which samples to keep. The selected samples must then be preserved in a pristine condition for return to the ascent and Earth return vehicles.

Sample acquisition, analysis and preservation are enabling technologies which effect the science and exploration value of an autonomous rover. Similar to the two types of rover navigation scenarios, SAAP technology falls within two bounding cases. In one, operators on Earth retain full control. All directives concerning

sample selection, acquisition and analysis are controlled from Earth. In the other, the more ambitious scenario, these functions are performed autonomously based on planning and decision criteria formulated on Earth prior to the mission. During the mission, Earth operators interact with the system in a supervisory manner.

In developing SAAP technology, it will be important to determine the extent to which autonomous operations can be used to augment Earth-based control. SAAP systems will be operating in unfamiliar environments which can only be approximated on Earth. While there is significant reluctance among the scientific community to lessen Earth-based control of SAAP operations, it is recognized that much more science can be accomplished if more planning and decision making responsibility can be initially assigned or later delegated to the SAAP system.

The state-of-the-art is represented by Viking lander sample acquisition and analysis technology and Apollo / Space Shuttle biomedical freezer sample preservation technology. The Viking sample acquisition system was basically a scoop, with a movable lid and a backhoe hinged to its lower surface, attached to the end of a retractable boom. The analytical instruments included a x-ray fluorescence spectrometer, a gas chromatograph-mass spectrometer and a biology package.

Lunar samples returned to Earth in the Apollo program were not temperature controlled as will be required for Martian samples. Thermally controlled container technology, however, has been developed for biomedical experiments aboard the Space Shuttle.

The sample acquisition, analysis and preservation technology needs include:

- o High speed broadband multispectral data acquisition and analysis
- o Lightweight, low power manipulator(s), end effector(s) and tools with associated control system,
- o Autonomous core drilling,
- o Lightweight, low power analytical instruments and technology for elemental, chemical and physical property determination, and onboard analysis and decision making.
- o Sample preparation technology to process samples for analysis and/or storage,
- o Sample preservation technology including environmental control of samples (most importantly lowest mean annual temperature of about -40 degrees C) during in-situ analysis and for the duration of the mission.

#### **10. PATHFINDER PLANETARY ROVER AND SAMPLE ACQUISITION, ANALYSIS AND PRESERVATION DEVELOPMENT PLANS**

The Pathfinder Planetary Rover and Sample Acquisition, Analysis and Preservation programs are multi-year, focused development programs which will validate automated and piloted rover technological maturity sufficient for use by prudent project managers. The initial focus is on automated unmanned rover technology for exploration and science. Later needs are for rover systems for automated construction and mining and for exploration with human driven rovers.

The PPR program includes three major work element types; namely:

- o Advanced development of unmanned planetary rover subsystem technology,
- o An integrated unmanned planetary rover testbed,

- o Advanced development of piloted rover technology for exploration and automated rover technology for mining and construction.

Advanced unmanned planetary rover subsystem technology development encompasses seven work items; namely, mobility, navigation, ground operations, computation, power, temperature control and communications. The SAAP program will develop the subsystem technology required to identify, acquire, analyze and return to Earth scientifically valuable specimens from a planets surface or near subsurface. The technology requirements delineated in sections 2 through 9 will be achieved, in a phased manner, within a five (5) year time frame.

An integrated testbed will be defined, implemented and operated. This testbed will provide a focus for and a means of validating the advanced subsystem technology as well as the integration technology and will serve as a mechanism for the technology transfer process. The definition phase will be completed in FY 90. The initial testbed implementation will be completed in FY 92.

Piloted rover and automated mining/construction rover technology work elements are planned for initiation in the FY 91 time period. Detailed planning of these work elements has not yet been performed.

NASA responsibility for the PPR program rests with the Information Sciences and Human Factors Division of OAST. NASA responsibility for the SAAP program rests with the Materials and Structures Division of OAST. JPL has coordination and management responsibility for the implementation of both the PPR and SAAP programs. Other NASA centers involved or to be involved in the program include the Ames, Lewis and Langley Research Centers and Johnson Space Center. Universities will have an important role in the PPR program as well. Carnegie Mellon University (CMU) is developing an innovative legged locomotion mobility prototype vehicle with its associated sensing, perception, planning and reasoning systems. Industry will also play an important role in the PPR program; initially in the component development area and later, in the implementation and operation of the integrated testbed.

## 11. SUMMARY

Planetary rover technology advances are needed in the areas of mobility, navigation, sample acquisition, analysis and preservation, mission operations, computation, power, temperature control and communication technologies. The NASA Office of Aeronautics and Space Technology (OAST) is addressing these technology needs in its base research and development program, the Civil Space Technology Initiative (CSTI) and a new technology initiative entitled "Pathfinder". The Pathfinder Planetary Rover and sample Acquisition, Analysis and Preservation programs will develop and validate the technologies needed for both robotic and piloted exploration of various planetary surfaces. Pathfinder does not represent, in itself, a commitment to any particular mission. It will, however, provide a variety of high-leverage technologies which will help enable future national decisions regarding exploration of the Solar System.

## ACKNOWLEDGEMENT

This paper presents the results of one phase of research carried out at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration.

## RICE-OBOT I: AN INTELLIGENT AUTONOMOUS MOBILE ROBOT\*

R. deFigueiredo, L. Ciscen, D. Berberian

Department of Electrical and Computer Engineering  
Rice University  
Houston, Texas 77251-1892

### Abstract

The *Rice-obot I* is the first in a series of Intelligent Autonomous Mobile Robots (IAMRs) being developed at Rice University's Cooperative Intelligent Mobile Robots (CIMR) lab. The *Rice-obot I* is mainly designed to be a testbed for various robotic and AI techniques, and a platform for developing intelligent control systems for exploratory robots. In this paper, we present the need for a generalized environment capable of combining all of the control, sensory and knowledge systems of an IAMR. We introduce *Lisp-Nodes* as such a system, and we develop the basic concepts of nodes, messages and classes. Furthermore, we show how the control system of the *Rice-obot I* is implemented as sub-systems in *Lisp-Nodes*.

### 1. Introduction

The *Rice-obot I* is the first in a series of Intelligent Autonomous Mobile Robots (IAMRs) being developed at Rice University's Cooperative Intelligent Mobile Robots (CIMR) lab. Rice University has developed strong relationships with several groups at the Johnson Space Center, and thus the mobile robotics program has emphasized technologies applicable to Space Robotics and exploratory roving vehicles. The *Rice-obot I* is mainly designed to be a test platform on which various control, hardware and AI concepts can be easily inserted and tested. Also, we are interested in developing onboard intelligent command systems required for autonomous exploratory robots. Furthermore, we want the robot to be able to perform repairs and maintenance on objects, as a space robot might do to a satellite. To achieve these goals, *Rice-obot I* was designed to include several advanced capabilities. Among these, the most important are:

A. To be totally autonomous. This means that all of the computing is onboard, the system uses radio links to the basestation (thus no cables), and it has an onboard power system.

---

\*Supported by the NASA Grant NAG-9-208, the State of Texas Grant TATP 2982, and a grant from Texas Instruments

B. To be able to navigate in an unstructured environment. By unstructured we mean that an incomplete (or nonexistent) map of the area is given. This implies that it should possess map-making capabilities.

C. To incorporate advanced sensors, including a laser 3-D mapper and stereo vision.

D. To communicate with users at the basestation at a high level; this maximizes the amount of information transmitted and minimizes the data bandwidth of the link.

E. It should incorporate dual dexterous arms and a set of tools for manipulation and assembly/disassembly of objects.

and, most importantly,

F. To be highly "modular" both in hardware and software, so that various AI systems can be integrated and interchanged relatively easily. This enables us to use already developed, high-level AI techniques from other sources with very little modification.

A fair amount of research has been done on the various areas associated with mobile robotics, including path planning (Weisbin[3], Thorpe[13], Meng[5,6]), sensor integration (Hirzinger[8], Harmon[12], and Thorpe[13]), and various people on topics from obstacle avoidance to advanced control algorithms. It is important to note that developers have relied on a large variety of AI systems as the basis for their intelligence including expert systems, neural nets, blackboard systems, and semantic nets. However, it has become apparent that no single knowledge representation scheme is sufficient for dealing with the myriad of different tasks, situations, and objects encountered by an IAMR. In that light, various researchers have developed different schemes for integrating the various AI and control system ([2], [4], [10], [16], [18]). It is even more complicated to make such systems "modular" (although Brooks' system is a good example). We came to the conclusion that we needed a fundamental, underlying environment in which we could install several different AI techniques concurrently, and that could easily handle the multitasking and multiprocessors in an IAMR. Thus, we have developed the Lisp-Nodes environment. In Lisp-Nodes, all of the knowledge and control of the robot is embodied as systems of nodes. The node network is divided into major and minor sub-systems which communicate with one another through a high-level protocol. With this system, we can replace pieces of the overall system without affecting the remaining parts, and it can expand in an organized and controlled manner.

## 2. Hardware Overview

The robot includes an intelligent base, two robotic arms, several on-board computers, a



stereo vision system, a 3-D laser mapper and an Ultrasonic system (see following figures). The robot is mounted on a commercially available mobile base which contains its own processor and power. The rest of the components mount onto a tubular steel frame which, in turn, bolts to the frame of the base. A standard VME bus is used. To augment this, there is a board-to board bus connecting the vision equipment for high-speed picture transfer. The main processor is a 68030 based computer made by LYNX Systems. The processor runs a real-time version of UNIX especially designed for control of devices such as robotic arms. The motors on the robot, including those on both arms, the Z-tables and the pan/tilt/aim of the cameras, are controlled (through the UNIX processor) by two servo controller cards on the VME bus. The two robots arms each have five degrees of freedom; they have the advantage over other arms that they are very light (15 lbs. apiece), yet they have high accuracy (approx. 1/5 mm. with modifications we have made) and a relatively high payload (5 lbs.). The arms mount on the Z-tables, thus adding an additional degree of freedom. The instrument pod mounts on a pan/tilt unit connected on the top of the robot. The four major items on the pod consist of two cameras, a laser ranging system, and one ultrasonic ranging system. The vision system is augmented by real-time processing boards that perform low-level processing of the image, separate the image into "blobs", and pass the information to the UNIX processor. The robot communicates via radio frequency RS232 to a windowing workstation, which acts as the user interface. A special LISP processor, the TI Explorer II, is also mounted on-board the robot to handle most of the high-level Artificial Intelligence tasks. The Explorer contains a multiprocessor Odyssey board for general high-speed signal processing. The Odyssey acquires high-speed vision data through an extension to the high-speed vision bus. Most of the low-level control and command of the robot occurs in the UNIX processor. For the most part, the two computers communicate using an Ethernet link. This on board network is especially useful during debugging, because it can be easily connected to the major on-campus computer network.

### 3. Lisp-Nodes Introduction

Lisp-Nodes is based upon *nodes*. A node can embody many different concepts. It could be a single if/then expression in an Expert System, or a cell in a Neural Net. Alternately, one node could embody all of the low-level vision processing for a 3-D vision system. The difference between the former two examples and the latter can be viewed as the amount of coupling; that is, the former is loosely coupled, the latter is tightly coupled. Both types of coupling are needed on a mobile robot. Some sub-systems such as the kinematics of the arm, work best as a tightly coupled routine. Others, like the global knowledge base, need a loosely coupled network of nodes, while the vision system needs some of both. Lisp-Nodes allows all levels of coupling within a uniform environment. It is important to note that Lisp-Node's proprioceptive knowledge is mostly at the node and interconnection level; i.e. the system learns by creating and destroying nodes and their connections. Thus the

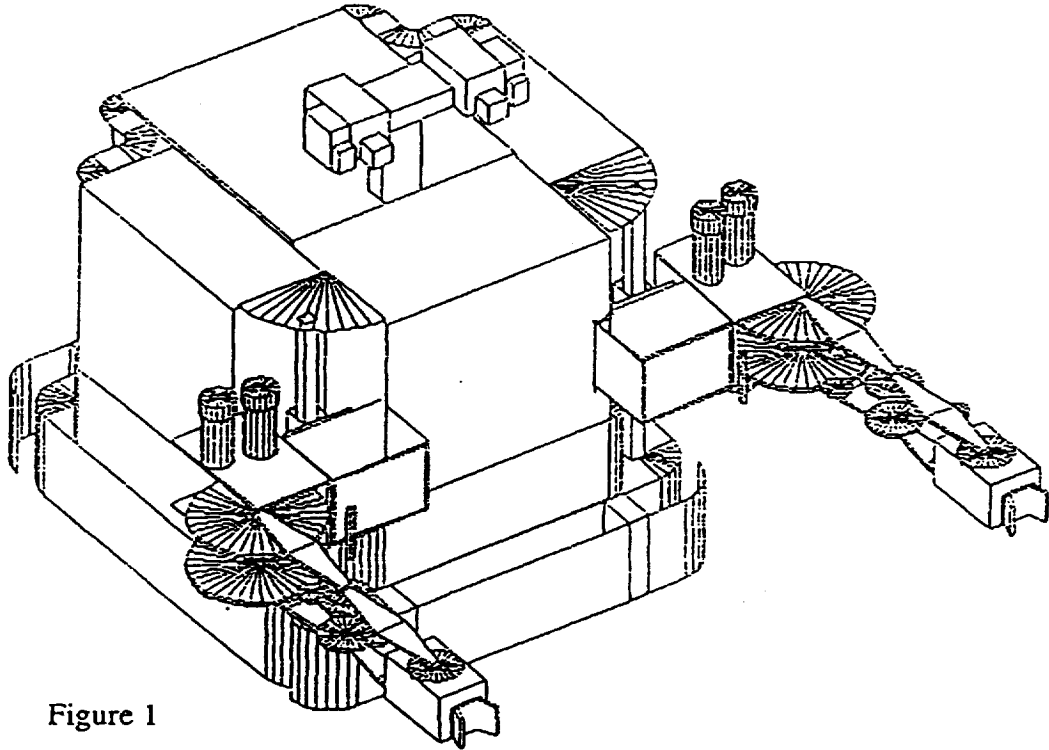


Figure 1

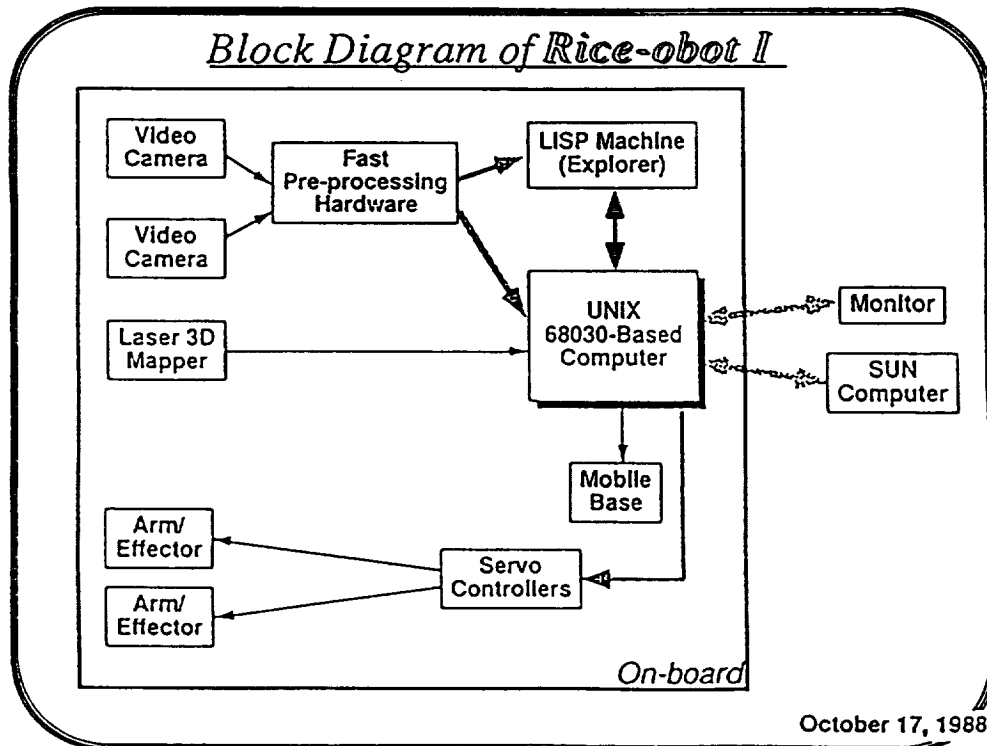


Figure 2

more tightly coupled the sub-system, the harder it is to be taught new techniques.

Another important aspect of nodes is that each node runs concurrently; so different groups of nodes can be processing different information simultaneously. This is critical since many different systems on the robot need to process information at the same time. For example, some nodes could be processing vision data while others plan a path.

Nodes communicate with each other using messages. Messages in Lisp-Nodes act just like messages in a computer network; they can send packets of information between any two nodes. The message packets can contain any information ranging from a string to a complex LISP expression. Each node can respond to many messages, and messages can be added and deleted from its capabilities. By modifying what messages are sent and received, the interconnection of nodes is also changed.

Another important capability of LispNodes is its ability to group nodes together; a group (or *class*) could be all nodes with a specific property, or all nodes pertaining to a particular subsystem. One example usage of classes is for Minsky's frame concept. Grouping enables an arbitrary node to communicate with a whole class of nodes without necessarily knowing which nodes are in the class; thus classes act very much like blackboards. Classes are formed by creating a *class-node* that processes and passes messages to all nodes within the group. Figure 3 shows a class node distributing a message.

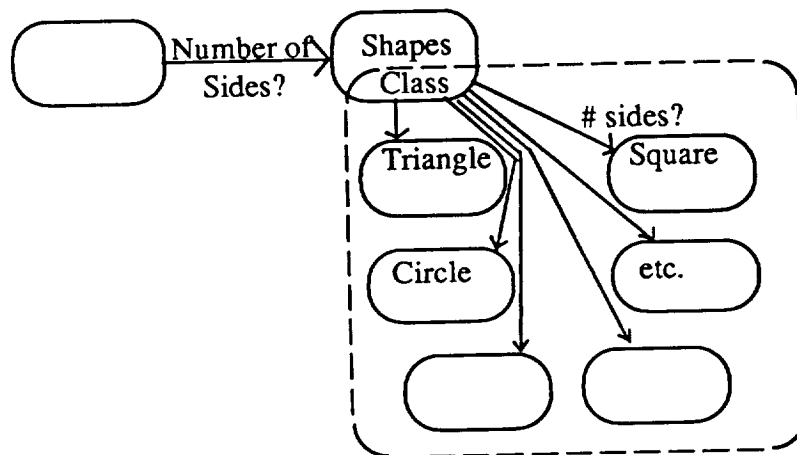


Figure 3

#### 4. Subsystem Organization

As has been mentioned, all of the sensor, control, and knowledge representation systems on the robot are connected to the Lisp-Nodes environment. This, in itself, does not impose any order on the system. The control system organized on top of the nodes has to be structured enough to

enable the robot to repeatably and predictably perform complicated functions, yet flexible enough to be easily modifiable. Thus we chose a knowledge-base driven system consisting, at the top level, of a very few distinct major subsystems which communicate and interact using a fairly simple set of rules. These subsystems are implemented using the classes in Lisp-Nodes, and thus act like mini-blackboards. These main subsystems are further divided into sub-subsystems as necessary. The sub-subsystems act the same as the main systems, except for two main differences: a) in these sub-subsystems, classes can overlap (i.e. a node can be in several different sub-subsystems simultaneously), and b) they can be created and destroyed. Whenever a node or class is created/destroyed, the classes it belongs to are informed of the change; these classes, in turn, can inform their parent-classes of the change, which inform the next higher level, etc. To ensure consistency, major subsystems (and their subsystems) route all requests for creating/destroying nodes that are contained in a different major subsystem through that system's class. The major subsystems that are being implemented on the Rice-obot are as follows:

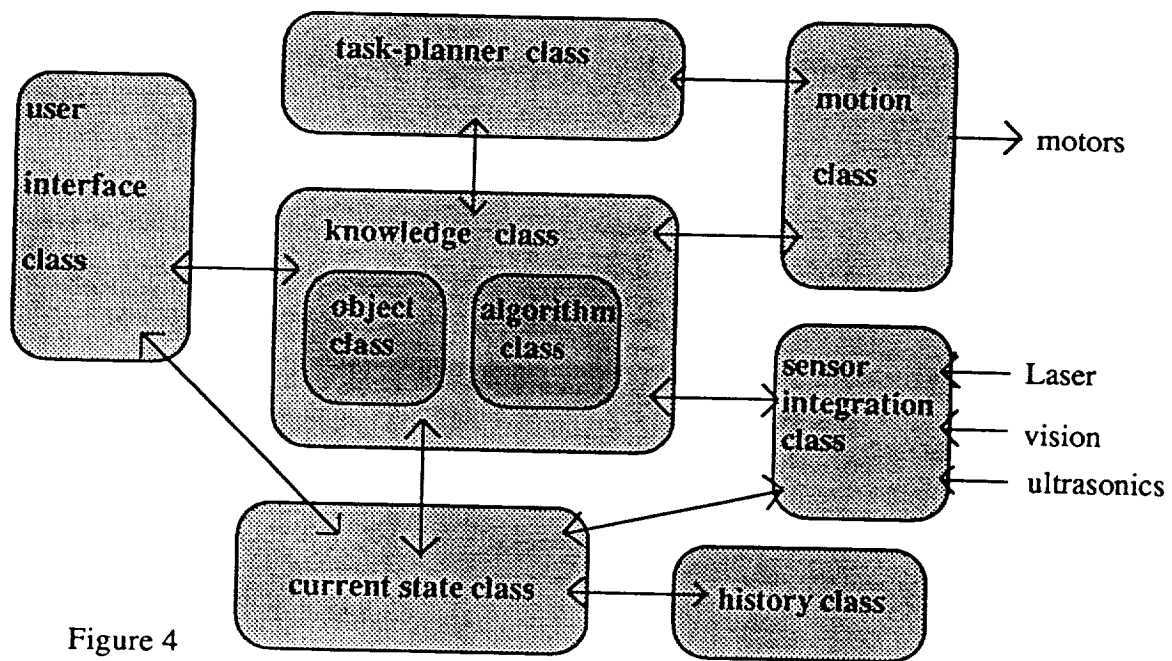


Figure 4

The central system in the robot is the knowledge-class. This system stores knowledge not only about objects and their interrelationships, but also about techniques for subdividing tasks, and the relationships/dependencies of tasks. The task-planner class is responsible for developing goals and sub-goals, for resolving conflicts, and for generating the most efficient plan. It queries the current-state class, the knowledge class, and the user-interface (when necessary) about how to perform its duties. In addition to maintaining a record of its "current" tasks, the task-planner can

develop *theoretical* plans and simulate their progression to determine the best plan. The user interface contains nodes for controlling and creating the environment that the human interactor sees. It controls simple commands/responses sent and it relays information as a debugger. Ultimately, it will also contain a meta-english language (i.e. a structured and simplified natural language) for high-level interaction. The current state class represents the robot's most up-to-date perception of its surroundings and its status. This class contains several different representations of the objects and relationships in its vicinity. These include polygonal representations of objects (mostly used by the path planner), dependency nets, solid models and abstract semantic nets. Depending on which subsystem is sending/receiving information to this class, different representation are presented in response. The history class is a *selective* history of the robot's actions and surroundings. The sensor-integration class deals with developing a complete model of the environment. It creates "sketches" from each sensor and integrates them to form a complete model. This model is passed to the current-state class to enhance its model. The integration-class uses information from the knowledge class to identify objects and to aid in the scheme construction. Finally, the motion-control class handles the obstacle-avoidance, low-level trajectory planning, and local path planning. It draws upon the current-state class for a map of the surroundings (which may be incomplete).

## 5. Results

The hardware systems on the Rice-obot I are nearing completion. Figure 5 shows the assembled Rice-obot I. The Explorer has been mounted onboard the robot, and the arms, pan/tilt, z-tables and superstructure of the robot are mounted. Only the onboard battery power system, some of the arm-control circuitry and the laser mapper are not yet built. We have a fully functional version of Lisp-Nodes running onboard, and it is interfaced to the low-level sensor and motion control subroutines. Furthermore, the major sub-system class nodes exist, and a preliminary set of rules for their interaction. We have also installed a local path-planner based upon the paper by Alex Meng[6]. Meng's path planner is functional on the Explorer, although it has not been fully integrated into the Lisp-Nodes environment. This path-planning system has already provided two useful results: a) it has demonstrated and tested the basic mobility and control of the robot and b) it has demonstrated the ability of the robot to easily attach developed subsystems. The main areas to be developed on the robot are the sensor integration subsystem and the current-state system. Furthermore, it is not yet clear what extensions (if any) need to be made in the structure of the interaction between subsystems to ensure consistency. This will become most critical as the current-state system is further developed, since it communicates heavily with other subsystems.

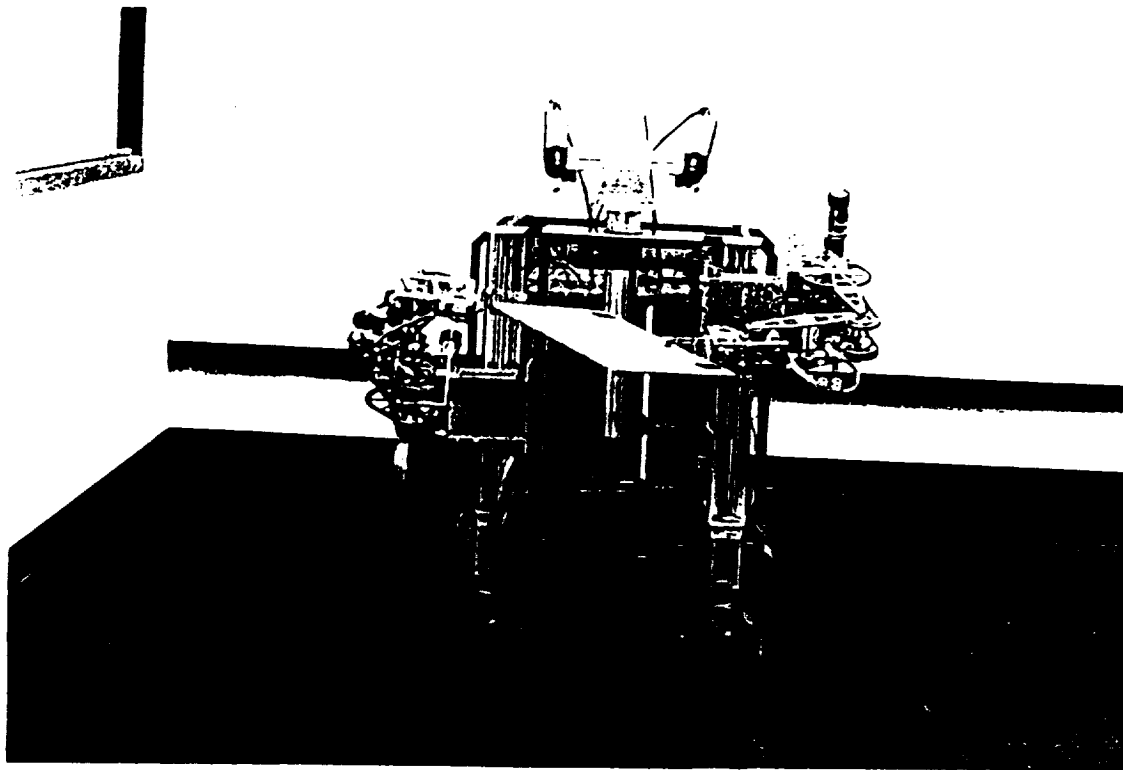


Figure 5

## 6. Significance towards Exploration & Space Robotics

One of the basic problems inhibiting development in intelligent robotics is the lack of a basic overall "operating system" for complex, multisensory mobile robots on which new concepts can be easily integrated with other already-developed pieces. Every time a new mobile robot is developed, the entire sensory/control/perception system has to be completely recreated. Furthermore, current intelligent control systems are very inflexible to modifications and cannot easily embody several different knowledge structures. Thus they tend to be "specialized" for a particular task; exploratory robots need a more broad-based control system. Lisp-Nodes provides the basis for such a control system. Furthermore, The Rice-obot I is a good tool for developing such concepts as Lisp-Nodes because of the advanced operating systems (UNIX and Explorer) onboard, because of the high bandwidth of data transfer among the subsystems, and because of the advanced sensors available. The Rice-obot I is the first step in the Rice University CIMR lab's goal of having multiple mobile robots cooperating in performing tasks in a real-world environment.

## 7. Bibliography

- [1] DeFigueiredo, R.J.P., Wang, K., *An "Evolving Frame" Approach to Learning with Application to Adaptive Navigation*, Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, Atlanta, Georgia, 1986.
- [2] Kohn, W., Skillman, T., *Hierarchical Control Systems for Autonomous Space Robots*, Boeing Electronics and Computer Service Company.
- [3] Weisbin, C., de Saussure, G., Kammer, D., *Self-Controlled- A Real-Time Expert System For an Autonomous Mobile Robot*, Computers in Mechanical Engineering, Vol.5 No. 2, Sept., 1986.
- [4] Isik, C., Meystel, A., *Pilot Level of a Hierarchical Controller for an Unmanned Mobile Robot*, IEEE Journal of Robotics and Automation, Vol.4, No.3, June, 1988.
- [5] Meng, A., *Free Space Modeling and Geometric Motion Planning Under Location Uncertainty*, Artificial Intelligence Lab, Texas Instruments.
- [6] Meng, A., *A Methodology of Map-Guided Autonomous Navigation with Range Sensor in Dynamic Environment*, Proceedings of the SPIE Conference on Mobile Robotics III, 1987.
- [7] Hayati, S., *Hybrid Position/Force Control of Multi-Arm Cooperating Robots*, IEEE Journal of Robotics and Automation, 1986.
- [8] Hirizinger, G. *Multisensory Robots and Sensor-Based Path Generators*, IEEE Journal of Robotics and Automation, 1986.
- [9] Harmon, S.Y., *A Technique for Coordinating Autonomous Robots*, IEEE Journal of Robotics and Automation, 1986.
- [10] Ahmad, S., *Real-Time Multiprocessor-Based robot Control*, IEEE Journal of Robotics and Automation, 1986.

- [11] Nayeswara Rao, S.V., *Concurrent Algorithms for Autonomous Robot Navigation in an Unexplored Terrain*, IEEE Journal of Robotics and Automation, 1986.
- [12] Harmon, S.Y., *Sensor data Fusion through a Distributed Blackboard*, IEEE Journal of Robotics and Automation, 1986.
- [13] Thorpe, C., *An Architecture for sensor Fusion in a Mobile Robot*, IEEE Journal of Robotics and Automation, 1986.
- [14] Parodi, A.M. *An Intelligent System for an Autonomous Vehicle*, IEEE Journal of Robotics and Automation, 1986.
- [15] Zacksenhouse, deFigueiredo, R.J.P., Johnson, D., *A Neural Network Architecture for Cue-Based Motion Planning*, to be presented at the 27th IEEE Conference on Decision and Control, Austin, Tx., Dec., 1988.
- [16] Brooks, R., *A Hardware Retargetable Distributed Layered Architecture for Mobile Robot Control*, MIT Artificial Intelligence Laboratory.
- [17] Flynn, A., Brooks, R., *MIT Mobile Robots- What's Next?* MIT Artificial Intelligence Laboratory.
- [18] Skillman, T., Kohn, W., *Blackboard Based Hierarchical Intelligent Control System*, AIAA Journal Guide Cont. and Dyn.
- [19] Nii, P., *Blackboard Systems: The Blackboard Model of Problem Solving and the Evolution of Blackboard Architectures*, The AI Magazine, Summer, 1986.



# Satellite-Map Position Estimation for the Mars Rover

Akira Hayashi

Department of Computer Science  
University of Texas at Austin  
Austin, Texas 78712

Thomas Dean<sup>1</sup>

Department of Computer Science  
Brown University  
Box 1910, Providence, RI 02912

## Abstract

This paper describes a method for locating the Mars rover using an elevation map generated from satellite data. In exploring its environment, the rover is assumed to generate a local rover-centered elevation map that can be used to extract information about the relative position and orientation of landmarks corresponding to local maxima. These landmarks are integrated into a stochastic map which is then matched with the satellite map to obtain an estimate of the robot's current location. The landmarks are not explicitly represented in the satellite map. The results of our matching algorithm correspond to a probabilistic assessment of whether or not the robot is located within a given region of the satellite map. By assigning a probabilistic interpretation to the information stored in the satellite map, we are able to provide a precise characterisation of the results computed by our matching algorithm.

## 1 Introduction

In the current projections for the Mars Rover project, a satellite is placed in Mars orbit prior to the rover's arrival in order to collect stereo images of the Martian surface with approximately 1 meter resolution. These images are relayed to earth and used to generate a high-resolution elevation map of the regions that the rover is expected to explore. Once the rover has landed on Mars, this elevation map will be used to keep track of the position of the rover and plan out paths for it to follow in performing its exploration of the planet's surface. As the rover moves about, it will use passive-stereo imaging and a laser range finder to construct a depth map of its immediate area. This

depth map is then converted into an elevation map which is merged with the map generated from satellite data to provide greater detail.

We will refer to the map generated from satellite data simply as the *satellite map*, and the map generated from local observations made by the rover as the *rover map*. In this paper, we describe a technique for merging these two maps by using the rover map to locate the rover's position in the satellite map. Our method requires that the rover be able to extract the location of *landmarks* from the rover map, where a landmark corresponds to a local maximum (or *peak*) in the surrounding terrain. We assume that the measurements made by the rover are subject to known errors. The relative locations of the landmarks with respect to the rover's current location are stored in *stochastic map* [Smith *et al.*, 1985] that is maintained using the *approximate transform* method of [Smith and Cheeseman, 1986] (see also [Durrant-Whyte, 1988]). We describe an algorithm that provides an estimate of the rover's current location in terms of a probability assignment to fixed regions in the satellite map. For our methods to work, the terrain must have such locally observable features that can be differentiated given the resolution and accuracy of the information stored in the satellite map. The work described in this paper represents a specific application of a general technique first described in [Hayashi and Dean, 1988].

## 2 Problem Definition

### 2.1 Satellite Map

In the satellite map, the area of interest is divided into small square regions of the same size referred to as *sectors*. For each sector, the map contains both upper ( $H^+$ ) and lower ( $H^-$ ) bounds on the elevation

<sup>1</sup>This work was supported in part by the National Science Foundation under grant IRI-8612644 and by the Advanced Research Projects Agency of the Department of Defense and was monitored by the Air Force Office of Scientific Research under Contract No. F49620-88-C-0132.

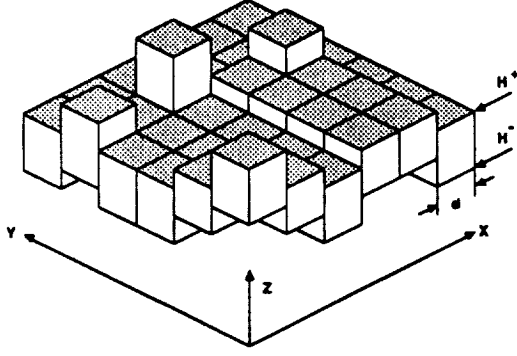


Figure 1: Satellite map

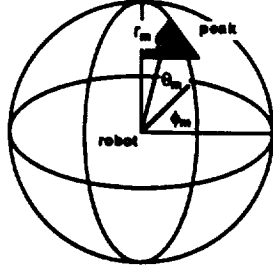


Figure 2: Polar coordinates for vision

within that sector (see Figure 1).

$$\begin{aligned} H^-_{i,j} &\leq Z(X,Y) \leq H^+_{i,j} \\ X_i &\leq X_{i+1}; X_i = i * d, X_{i+1} = (i+1) * d, 0 \leq i \leq N-1 \\ Y_j &\leq Y_{j+1}; Y_j = j * d, Y_{j+1} = (j+1) * d, 0 \leq j \leq N-1 \end{aligned}$$

where  $Z(X,Y)$  is the actual height at location  $(X,Y)$ ,  $d$  is the length of the side of a sector (also referred to as the map's *resolution*), and there are  $N^2$  sectors.  $H^-_{i,j}$  and  $H^+_{i,j}$  are the only information that the satellite map contains. There are no explicitly specified landmarks. The origin of the satellite map is chosen arbitrarily.

## 2.2 The Vision System

We assume that the rover has a vision system that can recognize the peaks of hills. The peaks of hills should be the most distinguishable features of a scene. We further assume that the vision system always succeeds in identifying unoccluded peaks in scenes, and that it is capable, with some statistical regularity, of identifying a peak as one that it has seen before. The vision system is not perfect, but the rover has a good approximation of its errors. The values that the vision system returns are the mean ( $v_m$ ) and

the standard deviation ( $\sigma_v$ ) of the peak's location in the rover-centered polar coordinate system shown in Figure 2.

$$v_m = \begin{pmatrix} r_m \\ \phi_m \\ \theta_m \end{pmatrix} \quad \sigma_v = \begin{pmatrix} \sigma_r \\ \sigma_\phi \\ \sigma_\theta \end{pmatrix}$$

where  $r$  is the distance to the peak,  $\phi$  is the azimuth angle to the peak in radians, and  $\theta$  is the elevation angle to the peak in radians. Azimuth angle is measured anti-clockwise from the East (e.g., 0 for East and  $\frac{1}{2}\pi$  for North), and the elevation angle is measured anti-clockwise from the horizontal direction. Obviously, a compass and gyro are necessary to make these measurements possible. We assume that each variable,  $r$ ,  $\phi$ , and  $\theta$ , forms a mutually independent normal distribution,  $N(r|r_m, \sigma_r^2)$ ,  $N(\phi|\phi_m, \sigma_\phi^2)$ ,  $N(\theta|\theta_m, \sigma_\theta^2)$ . The notation  $N(x|x_m, \sigma_x^2)$  is used as shorthand for a normal distribution of variable  $x$  with mean  $x_m$  and variance  $\sigma_x^2$ . The mean vector  $\mu_V(x,y,z)$  and the covariance matrix  $C_V(x,y,z)$  in Cartesian coordinate system will become necessary later and are derived from those in the polar coordinate system as:

$$\mu_V(x,y,z) = \begin{pmatrix} x_m \\ y_m \\ z_m \end{pmatrix} = \begin{pmatrix} r_m * \cos \theta_m * \cos \phi_m \\ r_m * \cos \theta_m * \sin \phi_m \\ r_m * \sin \theta_m \end{pmatrix} \quad (1)$$

$$C_V(x,y,z) = R_3 \cdot C_V(r,\phi,\theta) \cdot R_3^t \quad (2)$$

where

$$C_V(r,\phi,\theta) = \begin{pmatrix} \sigma_r^2 & 0 & 0 \\ 0 & \sigma_\phi^2 & 0 \\ 0 & 0 & \sigma_\theta^2 \end{pmatrix}$$

$$R_3 = \begin{pmatrix} x_m/r_m & -y_m & -z_m * \cos \phi_m \\ y_m/r_m & x_m & -z_m * \sin \phi_m \\ \sin \theta_m & 0 & r_m * \cos \theta_m \end{pmatrix}$$

$R_3$  is the value of  $J$  at the mean ( $\mu_V$ ), where  $J$  is the Jacobian of the transformation between the two coordinate systems. See Figure 3 for a visual characterization of the mean and covariance.

In the figures, we use *certainty ellipsoids* to represent the mean vectors and covariance matrices of our spatial variables. A certainty ellipsoid is the region within which its corresponding spatial variable lies given some probability (say 90%). The center of the ellipsoid is the mean vector. The relationship between a certainty ellipsoid and a covariance matrix is explained in [Smith and Cheeseman, 1986].

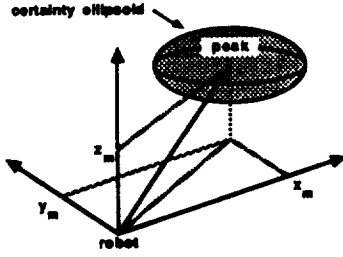


Figure 3: Returned values from the vision system

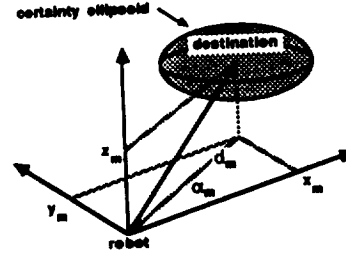


Figure 4: Movement coordinates

### 2.3 Rover's Movement

The format of the rover's movement command is a pair  $(d_m, \alpha_m)$  where  $d_m$  is the horizontal distance to a destination ( $m$ ), and  $\alpha_m$  is the azimuth angle to a destination in radians.

To consider movement errors, we assume actual movement  $(d, \alpha)$  is obtained using normal distributions,  $N(d|d_m, \sigma_d^2)$  and  $N(\alpha|\alpha_m, \sigma_\alpha^2)$  respectively. The means of the distributions ( $d_m$  and  $\alpha_m$ ) are the values specified in the movement command. The standard deviations ( $\sigma_d$  and  $\sigma_\alpha$ ) are computed from the accuracy of movements. The mean vector  $\mu_M(x, y, z)$  and the covariance matrix  $C_M(x, y, z)$  in Cartesian coordinate system are:

$$\mu_M(x, y, z) = \begin{pmatrix} x_m \\ y_m \\ z_m \end{pmatrix} = \begin{pmatrix} d_m \cdot \cos \alpha_m \\ d_m \cdot \sin \alpha_m \\ z_m \end{pmatrix} \quad (3)$$

$$C_M(x, y, z) = \begin{pmatrix} C_{xx} & C_{xy} & 0 \\ C_{yx} & C_{yy} & 0 \\ 0 & 0 & C_{zz} \end{pmatrix} \quad (4)$$

where

$$\begin{pmatrix} C_{xx} & C_{xy} \\ C_{yx} & C_{yy} \end{pmatrix} = R_2 \cdot C(d_m, \alpha_m) \cdot R_2^t$$

$$C(d_m, \alpha_m) = \begin{pmatrix} \sigma_d^2 & 0 \\ 0 & \sigma_\alpha^2 \end{pmatrix} \quad R_2 = \begin{pmatrix} x_m/d_m & -y_m/d_m \\ y_m/d_m & x_m/d_m \end{pmatrix}$$

$$C_{zz} = \sigma_z^2$$

See the illustration in Figure 4. Note that elevation angle to a destination is not specified in the movement command, since the rover can only move along the surface of the ground no matter whether it is uphill or downhill. Consequently the corresponding mean ( $z_m$ ) and the standard deviation ( $\sigma_z$ ) must be computed separately. They represent the expectation and the variance of the difference of height between

the current location and the destination, and therefore depend on the terrain along the movement path (which is not well known).

## 3 Building an Internal Map

When the rover explores an area, it builds an *internal map* so that it can match the internal map later with the satellite map. The internal map that we use is based on the *stochastic map* representation described in [Smith et al., 1985]. A stochastic map consists of a mean vector and a covariance matrix of spatial variables, and gives us estimates of the spatial relationships of these variables, their uncertainties, and their interdependencies. In addition, it provides us with a very elegant way of propagating constraints from various observations.

### 3.1 Rover's Internal Map

The internal map is a stochastic map which consists of a mean vector ( $\hat{u}$ ) and a covariance matrix ( $C(u, u)$ ) of the vector ( $u$ ) of the *spatial variables*.

$$u = \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{pmatrix} \quad \hat{u} = \begin{pmatrix} \hat{u}_1 \\ \hat{u}_2 \\ \vdots \\ \hat{u}_n \end{pmatrix}$$

$$C(u, u) = \begin{pmatrix} C(u_1, u_1) & C(u_1, u_2) & \cdots & C(u_1, u_n) \\ C(u_2, u_1) & C(u_2, u_2) & \cdots & C(u_2, u_n) \\ \vdots & \vdots & \ddots & \vdots \\ C(u_n, u_1) & C(u_n, u_2) & \cdots & C(u_n, u_n) \end{pmatrix}$$

where

$$u_i = (x_i, y_i, z_i)^t$$

$$\hat{u}_i = E(u_i)$$

$$C(u_i, u_j) = E((u_i - \hat{u}_i) \cdot (u_j - \hat{u}_j))$$

Our spatial variables ( $u_i$ 's) are

- locations of sensed peaks
- the rover's previous locations<sup>2</sup>
- the rover's current location ( $u_R$ )

The coordinate system used is a Cartesian coordinate system whose origin is the initial location of the rover ( $u_1$ ) and whose  $x, y, z$  axes are parallel to those of a satellite map. It is referred to as the *global coordinate system*.

### 3.2 Information in the Internal Map

The estimation of a spatial variable  $u_i$  (i.e. the estimation of the  $i$ th variable's coordinates in the global coordinate system) is  $\hat{u}_i$  and its uncertainty is obtained from  $C(u_i, u_i)$ . As a special case, for the initial location which was chosen as the origin of the global coordinate system,  $\hat{u}_1 = 0$ , and  $C(u_1, u_1)$  is a 0 matrix, since there is no uncertainty of  $u_1$  with respect to itself. The estimation of the spatial relationship between  $u_i$  and  $u_j$  is  $\hat{u}_j - \hat{u}_i$ , and its interdependency is obtained from  $C(u_j, u_i)$ .

### 3.3 Moving

When the rover makes a movement of  $u_{RR'}$  from its current location  $u_R$ , new location  $u_{R'}$  in the global coordinate system is

$$u_{R'} = u_R + u_{RR'}$$

The mean vector and the covariance matrix of  $u_{RR'}$  are

$$\hat{u}_{RR'} = \mu_M(x, y, z)$$

$$C(u_{RR'}, u_{RR'}) = C_M(x, y, z)$$

$\mu_M(x, y, z)$  and  $C_M(x, y, z)$  are defined in (3) and (4).

The rover's internal map is expanded from  $(\hat{u}, C(u, u))$  to  $(\hat{u}', C(u', u'))$  as follows (also see Figure 5):

$$\begin{aligned} \hat{u}_{R'} &= \hat{u}_R + \hat{u}_{RR'} \\ \hat{u}' &= \begin{pmatrix} \hat{u} \\ \hat{u}_{R'} \end{pmatrix} \\ C(u_{R'}, u_{R'}) &= C(u_R, u_R) + C(u_{RR'}, u_{RR'}) \\ C(u, u_{R'}) &= C(u, u_R) \\ C(u', u') &= \begin{pmatrix} C(u, u) & C(u, u_{R'})^t \\ C(u, u_{R'}) & C(u_{R'}, u_{R'}) \end{pmatrix} \end{aligned} \quad (5)$$

<sup>2</sup>In [Smith et al., 1985], the stochastic map does not contain the rover's previous locations. They are not necessary for navigation purposes. But the previous locations are useful for our matching purpose, since they give us more clues. From the view point of matching, there is no difference between peaks of hills and the rover's locations.

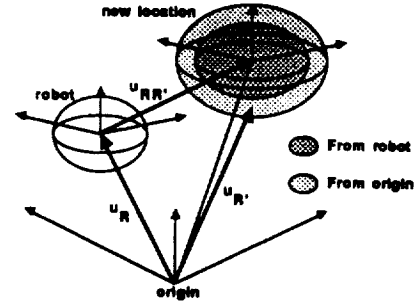


Figure 5: Rover movement

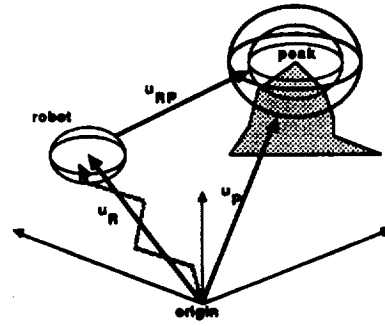


Figure 6: Seeing a new peak

### 3.4 Seeing a New Peak

When the rover finds a new peak at  $u_{RP}$  from its current location  $u_R$ , then the location of the peak  $u_P$  (in the global coordinate system) is

$$u_P = u_R + u_{RP}$$

The mean vector and the covariance matrix of  $u_{RP}$  are

$$\hat{u}_{RP} = \mu_V(x, y, z)$$

$$C(u_{RP}, u_{RP}) = C_V(x, y, z)$$

$\mu_V(x, y, z)$  and  $C_V(x, y, z)$  are defined in (1) and (2). The rover's internal map is expanded from  $(\hat{u}, C(u, u))$  to  $(\hat{u}', C(u', u'))$  in the same way as moving (see Figure 6).

### 3.5 Seeing the Same Peak Again

When the rover sees a peak from  $u_R$  and identifies it as  $u_P$  which it has seen before, the internal map is updated to get a better estimate of the spatial variables (Figure 7). In this case, the size of the map does not change, since no new spatial variable is added.

First, we define a sensor model as follows.

$$u_{RP'} = f(u) + v = u_P - u_R + v$$

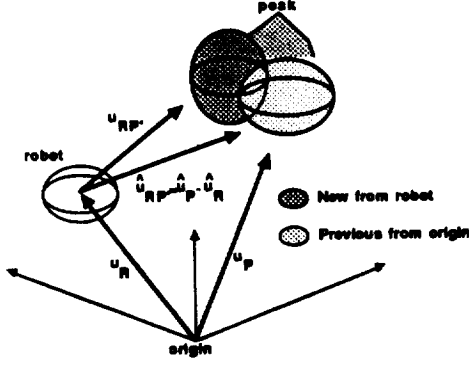


Figure 7: Seeing the same peak again

where  $u_{RP'}$  is the sensor measurement values, defined as  $\mu_V$  in (1),  $u$  is the vector of spatial variables, and  $v$  is the sensor noise with 0 mean.

Next, we compute the conditional estimates of the above sensor values ( $\hat{u}_{RP'}$ ), and their uncertainties ( $C(u_{RP'}, u_{RP'})$ ).

$$\hat{u}_{RP'} = f(\hat{u}) = F_u \cdot \hat{u} = \hat{u}_P - \hat{u}_R$$

$$C(u_{RP'}, u_{RP'}) = F_u \cdot C(u^-, u^-) \cdot F_u^t + C(v, v)$$

$$F_u = \delta f / \delta u = \begin{pmatrix} 0 & \dots & 0 & 1 & 0 & \dots & 0 & -1 & 0 & \dots & 0 \\ & & & u_P & & & & u_R & & & \end{pmatrix}$$

where  $C(v, v)$  is defined as  $C_V$  in (2).

Then, we update the map ( $\hat{u}^-, C(u^-, u^-)$ ) to ( $\hat{u}^+, C(u^+, u^+)$ ) using Kalman Filter equations [Smith and Cheeseman, 1986].

$$K = C(u^-, u^-) \cdot F_u^t \cdot [C(u_{RP'}, u_{RP'})]^{-1}$$

$$\hat{u}^+ = \hat{u}^- + K \cdot (u_{RP'} - \hat{u}_{RP'})$$

$$C(u^+, u^+) = C(u^-, u^-) - K \cdot F_u \cdot C(u^-, u^-)$$

Through the above equations we can get better estimates and certainties of not only  $u_P$  and  $u_R$ , but also all spatial variables that are correlated with  $u_P$  and  $u_R$ .

## 4 Matching the Two Maps

We have described both the satellite map and the rover's internal map. Note that the rover's internal map is built from observations independent of the satellite map. In this section we will explain the algorithm used to match the two maps in order to locate the rover with respect to the satellite map. The basic idea is the following. If we know the location of any of the spatial variables with respect to the satellite map, we can transform all spatial constraints between the two maps. It is then easy to compare the two maps. Since we don't know (with certainty) the location of

any spatial variable with respect to the satellite map, we attempt to rule out those locations that are implausible returning the likely locations as an estimate of the rover's location.

### 4.1 Sector Assignment

We start by assuming that the rover is located within certain sectors<sup>3</sup> of the satellite map.

$$\begin{aligned} X_i &\leq x_R < X_{i+1} \\ Y_i &\leq y_R < Y_{i+1} \\ H_{i,j}^- &\leq z_R \leq H_{i,j}^+ \end{aligned} \quad (7)$$

where  $(x_R, y_R, z_R)$  is the rover's current location in the internal map,  $(X_i, Y_j)$  is the vertex of  $(i, j)$ th sector in the satellite map,  $(X_{i+1}, Y_{j+1})$  is the vertex of  $(i+1, j+1)$ th sector in the satellite map, and  $H_{i,j}^-, H_{i,j}^+$  are the lower and upper bounds of heights in the  $(i, j)$ th sector in the satellite map.

### 4.2 Assignment as a Sensor Measurement

We try to express the assignment (7) as a kind of sensor measurement so that we can incorporate it into the internal map. From (7) we get

$$\begin{aligned} X_0 - X_{i+1} &\leq X_0 - x_R < X_0 - X_i \\ Y_0 - Y_{j+1} &\leq Y_0 - y_R < Y_0 - Y_j \\ Z_0 - H_{i,j}^+ &\leq Z_0 - z_R \leq Z_0 - H_{i,j}^- \end{aligned} \quad (8)$$

where  $(X_0, Y_0, Z_0)$  is the origin of the satellite map.

We choose  $(X_0, Y_0, H_{0,0}^-)$  as the origin. Note that the middle terms in (8) correspond to  $u_{RO}$ , the relative position of the satellite map's origin from the current location (Figure 8).

$$u_{RO} = u_0 - u_R \quad (9)$$

where  $u_0$  is the origin of the satellite map in the global coordinate system.

In our model, the vision system is supposed to return the mean vector and the covariance matrix for a sensed object. In other words, the vision system returns its certainty ellipsoid which corresponds to some confidence threshold. Hence we approximate the cuboid region (8) with a circumscribed ellipsoid (Figure 9), and then convert it to a mean vector and a covariance matrix. We can obtain a certainty ellipsoid from a covariance matrix [Smith and Cheeseman, 1986]. Here we follow the derivation in the opposite direction, and we get

$$\hat{u}_{RO} = \begin{pmatrix} x_m \\ y_m \\ z_m \end{pmatrix} = \begin{pmatrix} 0.0 - (i + 0.5)d \\ 0.0 - (j + 0.5)d \\ H_{0,0}^- - (H_{i,j}^- + H_{i,j}^+) * 0.5 \end{pmatrix} \quad (10)$$

<sup>3</sup>Sectors are the square regions that make up satellite maps.

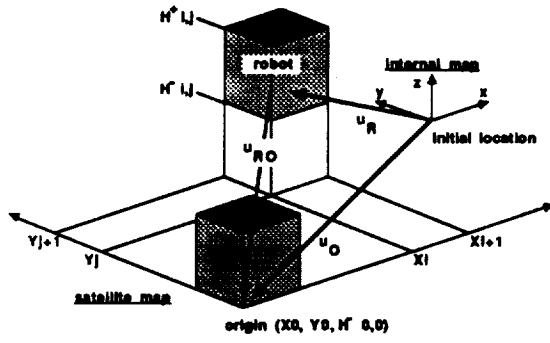


Figure 8: Assigning the rover a sector

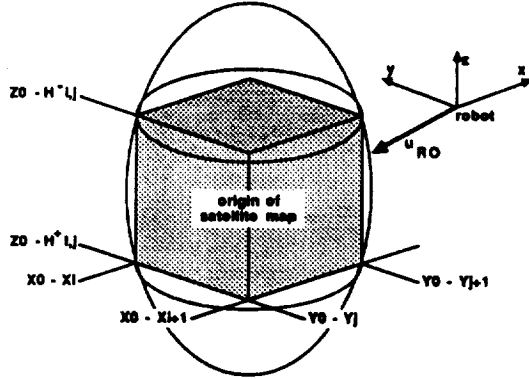


Figure 9: Circumscribed ellipsoid for  $u_{RO}$

$$C(u_{RO}, u_{RO}) = \begin{pmatrix} 3/4d^2 & 0 & 0 \\ 0 & 3/4d^2 & 0 \\ 0 & 0 & 3/4h^2 \end{pmatrix} \div \gamma^2 \quad (11)$$

where  $d$  is the resolution of the satellite map,  $h$  is  $(H^+_{i,j} - H^-_{i,j})$ , and  $\gamma$  in (11) is a constant chosen for a particular confidence threshold ( $prob^4$ ), and satisfies the following formula,

$$prob = 2 * \Phi(\gamma) - 1 - \sqrt{2/\pi} * \gamma * e^{-\gamma^2/2}$$

where  $\Phi(\gamma)$  is the cumulative density function of the unit normal distribution.

### 4.3 Merging the Two Maps

We incorporate  $u_O$  in (9) (the origin of the satellite map) into the rover's internal map as a virtual landmark. The internal map is expanded from  $(\hat{u}, C(u, u))$  to  $(\hat{u}', C(u', u'))$  as follows, using (10) and (11).

$$\hat{u}_O = \hat{u}_R + \hat{u}_{RO}$$

<sup>4</sup>The probability that  $(x, y, z)$  falls within the circumscribed ellipsoid. As we want the same sized ellipsoid in the next checking step,  $prob$  should be the same as the value which is used in the checking step.

$$\hat{u}' = \begin{pmatrix} \hat{u} \\ \hat{u}_O \end{pmatrix} \quad (12)$$

$$C(u_O, u_O) = C(u_R, u_R) + C(u_{RO}, u_{RO})$$

$$C(u, u_O) = C(u, u_R)$$

$$C(u', u') = \begin{pmatrix} C(u, u) & C(u, u_O)^t \\ C(u, u_O) & C(u_O, u_O) \end{pmatrix} \quad (13)$$

Although we have incorporated only the origin of the satellite map, we actually have merged the two maps. Via  $u_O$ , we can transform any constraint in the internal map to the constraint in the satellite map and vice versa.

### 4.4 Checking Assignment Consistency

The assignment made in (7) is arbitrary. We have to check whether it is consistent with both the given satellite map and the rover's observations. For any peak the rover has seen (and also for any previous location of the rover), its coordinates in the *satellite map's* coordinate system are given as

$$u'_P = u_P - u_O \quad (14)$$

If the initial assignment (7) is correct, then  $u'_P$  should be contained in some  $CUBOID(k, l)$  of the satellite map, for  $u'_P = (x'_P, y'_P, z'_P)^t$  is the coordinates of a peak in the satellite map's coordinate system. For some  $k$  and  $l$ , we have

$$\begin{aligned} X_k &\leq x'_P < X_{k+1} \\ Y_l &\leq y'_P < Y_{l+1} \\ H^-_{k,l} &\leq z'_P \leq H^+_{k,l} \end{aligned} \quad (15)$$

In order to check the inequalities (15), we need the actual distribution of  $u'_P$ . We use a certainty ellipsoid for that purpose. Given a confidence level ( $prob$ ), the certainty ellipsoid  $ELLPS(prob)$  for  $u'_P$  can be computed from its mean and covariance. These are obtained from the expanded internal map (12) and (13),

$$\hat{u}'_P = \hat{u}_P - \hat{u}_O$$

$$C(u'_P, u'_P) = C(u_P, u_P) + C(u_O, u_O) - 2 \cdot C(u_P, u_O)$$

When we set the confidence level ( $prob$ ) sufficiently large, we should expect

$$u'_P \in ELLPS(prob) \quad (16)$$

From (15) and (16), it follows that for any  $u_P$  in the internal map, there is some  $(k, l)$  s.t.

$$CUBOID(k, l) \text{ intersects } ELLPS(prob) \text{ of } u'_P. \quad (17)$$

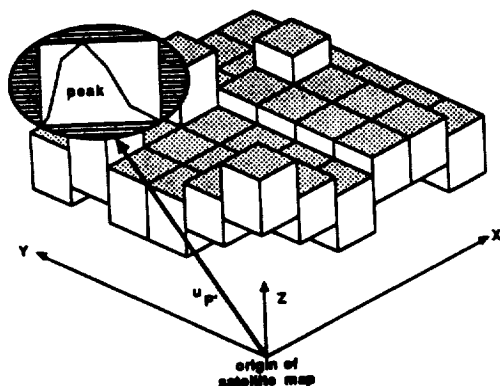


Figure 10: Checking consistency

Figure 10 illustrates this property.

To reiterate, statement (17) is only a necessary condition for a sector assignment (7) to be correct<sup>5</sup>. It is not a sufficient condition. In general, several sector assignments may satisfy (17). The total number of assignments which satisfy (17) depends on the satellite map's fuzziness and how many useful observations the rover has obtained so far. To find all sectors which are consistent, we repeat the above assignment and check steps for every sector in the satellite map.

## 5 Simulation

We tested our method using a simulation, and the initial results are encouraging. The simulation was carried out using a small terrain model to expedite the experiments. Realistic terrain data was obtained using techniques derived from fractal geometry. We assume that the rover has a vision system that can sense the peaks of hills.

The results of matching were weak at first. Too many sectors were left as possible locations of the rover. This was because sector assignments correspond to a much bigger certainty ellipsoid than those of sensor measurements; even an accurate sensor measurement became a vague one when it was compounded with a sector assignment in the matching phase. By making the covariance matrix of the sector assignment smaller, we were able to obtain satisfactory narrowing of the possible locations of the rover.

## 6 Conclusions

A method has been developed to locate the rover using local observations and a global satellite map. The

<sup>5</sup>To be more precise, it is not even a necessary condition because there is a slight chance that the actual position of a peak falls outside of the ellipsoid.

method provides answers to questions of the form: "Is it consistent to assume that the rover is located within a fixed area?" Its theoretical foundations are firm in the sense that the matching algorithm checks mathematically necessary conditions for a location assumption to be correct; the algorithm does not rely on any heuristics.

It should be noted that our problem cannot be handled using methods adapted from work on cruise missile guidance systems [Kober *et al.*, 1979]. Since cruise missiles are equipped with a highly accurate inertial guidance system, there is little uncertainty about their positions and orientations. More importantly, the missile sensors provide measurements from roughly the same perspective used in constructing the navigation map (the analog of our satellite map). We have also considered the possibility of using template matching techniques for our problem [Thorpe, 1981], but the low resolution of the satellite map makes landmark identification difficult and would appear to severely reduce the accuracy of the matching method.

## 7 Future Work

The simulations carried out so far are not sufficient. In order to make our method more effective and robust:

- We need guidelines on good threshold values for consistency checking.
- We need a better way of modeling of sector assignments than pseudo-sensor readings.
- We need to provide some way of tuning our method to suit the requirements of particular satellite maps and sensors.

So far, our main concern has been finding the current location of the rover with respect to the satellite map. But our matching method can be applied to the rover's navigation problem as well. Our work attacks the same problem as [Levitt *et al.*, 1987], but with the emphasis on metric rather than qualitative matching. In navigation problems, global maps usually contain some distinctive places (landmarks) and the paths are specified in terms of landmarks. Being able to determine the current location of the rover, identify landmarks, and determine the rover's location relative to a particular landmark are important problems that have to be solved for successful path planning and path following. Our matching algorithm should be useful in solving these problems since it can check the consistency of assumptions on the location of any of the observed peaks in the global (satellite) map, or on the relative location of the rover to some landmark.

## References

- [Durrant-Whyte, 1988] Hugh F. Durrant-Whyte. *Integration, Coordination and Control of Multi-Sensor Robot Systems*. Kluwer Academic Publishers, 1988.
- [Hayashi and Dean, 1988] Akira Hayashi and Thomas Dean. Locating a mobile robot using local observations and a global satellite map. In *Proceedings of Third IEEE International Symposium on Intelligent Control*, 1988.
- [Kober et al., 1979] W. Kober, C. Grosch, and T. Rosenthal. Three dimensional surface shell correlator. *SPIE*, 186, 1979.
- [Levitt et al., 1987] Tod S. Levitt, Daryl T. Lawton, David M. Chelberg, and Philip C. Nelson. Qualitative landmark-based path planning and following. In *Proceedings AAAI-87*, pages 689-694. AAAI, 1987.
- [Smith and Cheeseman, 1986] Randall Smith and Peter Cheeseman. On the representation of and estimation of spatial uncertainty. *The International Journal of Robotics Research*, 5:56-68, 1986.
- [Smith et al., 1985] Randall Smith, Mathew Self, and Peter Cheeseman. Estimating uncertain spatial relationships in robotics, 1985.
- [Thorpe, 1981] C. Thorpe. Sonar image processing - an application of template matching through relaxation. Technical Report CMU-RI-TR-81-6, Carnegie-Mellon University Robotics Institute, 1981.



**ROBOTIC SAMPLING SYSTEM FOR AN  
UNMANNED MARS MISSION**

**WENDELL CHUN  
MARTIN MARIETTA ASTRONAUTICS GROUP  
SPACE SYSTEMS COMPANY**

**ABSTRACT**

Major robotics opportunity for NASA will be the Mars Rover/Sample Return Mission which could be launched as early as the 1990s. The exploratory portion of this mission will include two autonomous subsystems: the rover vehicle and a sample handling system. The sample handling system is the key to the process of collecting Martian soils. This system could include a core drill, a general-purpose manipulator, tools, containers, a return canister, certification hardware and a labeling system. Integrated into a functional package, the sample handling system is analogous to a complex robotic workcell. This paper discusses the different components of the system, their interfaces, foreseeable problem areas and many options based on the scientific goals of the mission.

The various interfaces in the sample handling process (component to component and handling system to rover) will be a major engineering effort. Two critical evaluation criteria that will be imposed on the system are flexibility and reliability. It needs to be flexible enough to adapt to different scenarios and environments and acquire the most desirable specimens for return to Earth. Scientists may decide to change the distribution and ratio of core samples to rock samples in the canister. The long distance and duration of this planetary mission places a reliability burden on the hardware. The communication time delay between Earth and Mars minimizes operator interaction (teleoperation, supervisory modes) with the sample handler. An "intelligent" system will be required to plan the actions, make sample choices, interpret sensor inputs, and query unknown surroundings. A combination of autonomous functions and supervised movements will be integrated into the sample handling system.

**1. Introduction**

In the 1990s, robotic systems will be in operation in space and especially about the space station. The specific tasks include all forms of servicing, such as assembly, inspection, module changeout, refueling. However, there is a unique task that is not servicing, i.e. exploration. In particular, the Mars Rover/Sample Return mission promises to be a unique opportunity that could be launched in the 1990s. Exploration conjures up a sense of adventure and the unknown. This uncertainty separates a structured servicing task from an unstructured exploratory task. There are many issues that make this mission unique. This paper discusses the Mars mission, sampling hardware, sampling operations, and technical issues.

**2. Mars Mission**

It is man's inquisitive nature that drives him to explore the surface of Mars. Mars holds answers to many scientific questions about the origin of this universe. Refer to table 1 for some facts on Mars.

Table 1 Some Mars Facts

Planet Radius	3397.2 km (equator)
Mass	$6.418 \times 10^{23}$ kg
Bulk Density	$3.94 \text{ gm / cm}^3$
Gravitational Acceleration at the Surface	$3.73 \text{ m / sec}^2$ (0.38 Earth G)
Maximum Temperature	288 K (At Equator)
Minimum Temperature	150 K (Polar CO <sub>2</sub> )
Atmosphere	CO <sub>2</sub> , N <sub>2</sub> , Ar, O <sub>2</sub> , CO
Mean Pressure	(.089 psi)
Wind	2 to 7 m / sec
Dust Storms	1 or 2 / year

Mars is characterized by lava flows, crevices, boulder fields, mountains, dunes and craters. The terrain varies from drift material ranging in consistency from ordinary kitchen flour<sup>1</sup> to jagged rocks and boulders.

The scientific goals for planetary exploration are: (1) to understand how the solar system originated, (2) to understand how the planets evolved and to understand their present state, (3) to learn what conditions led to the origin of life, and (4) to learn how physical laws work in large systems.<sup>2</sup> Mars is the next focus because of its accessibility and relationship to Earth. By returning samples, their analysis will determine the chemical composition and mineralogy of materials from a selected region.

In particular, the absolute age of the rock and soil can be determined. Looking at a scene of continuous rocks (Fig. 1), it is hard to imagine that a system could differentiate and pick up a unique rock.

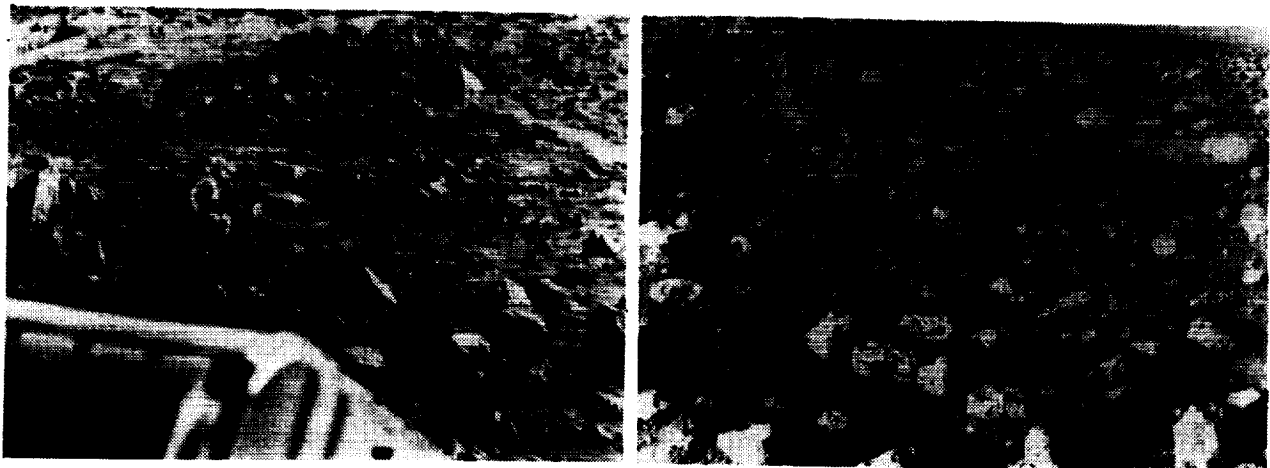


Figure 1 Photos of Rocks on the Martian Surface

### 3. Sampling Hardware

The functional steps of sample handing are outlined by JSC<sup>2</sup> and illustrated in Figure 2.

ORIGINAL PAGE IS  
OF POOR QUALITY

ORIGINAL PAGE  
BLACK AND WHITE PHOTOGRAPH

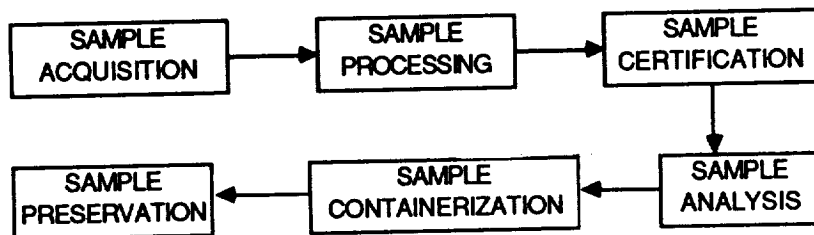


Figure 2 Sample Handling Functions

Sample acquisition is the physical interaction with the planet surface, e.g. picking up a rock or scooping loose soil. The captured sample then is transferred to processing. Processing prepares the sample for certification or analysis. In some cases, the rock sample might be too large for the test apparatus and needs to be split into smaller pieces. At other times, the soil is screened by a sieve.

Certification is the preliminary check to diagnose the character of a sample. At this point, the sample will be diagnosed from preliminary measurements to decide whether to return the sample to Earth or put it through analysis. Analysis entails further measurements and is the principal source of scientific information. The samples that are to be returned are packaged in containers and tagged with pertinent information such as location, temperature, humidity, etc. Sample preservation is the final step and protects the sample until it is received on Earth. A successful mission requires the specimen to arrive in pristine condition (no contamination, shock, or vibration damage).

The hardware for sampling is depicted in Figure 3. The subsystems shown include the manipulator, tools, and rover. The desired scientific samples dictate the tools required and the tools affect the container design that in turn drives the canister design.

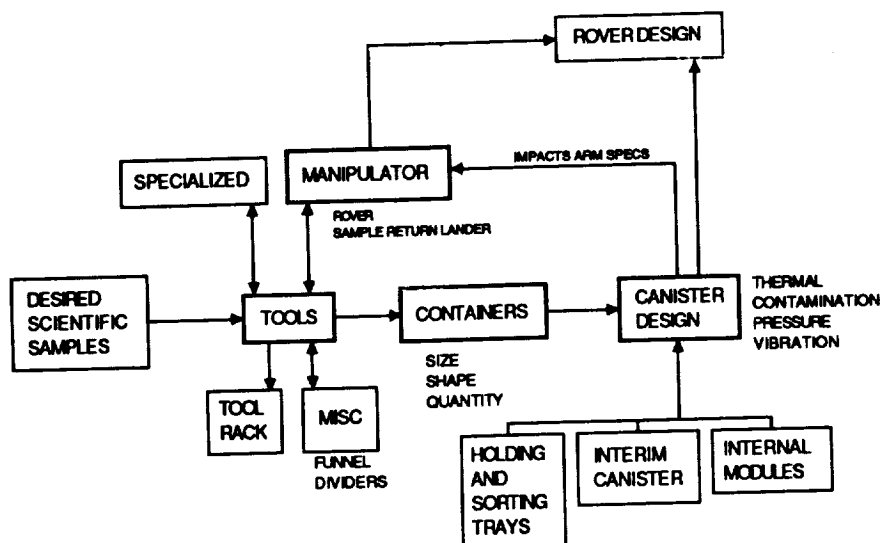


Figure 3 Hardware Interface

The current thinking is to have two manipulators: one on the rover and one on the lander. Both arms should be compatible with a common set of tools. The sample return lander manipulator is used for contingency samples at the beginning of the mission, while the second manipulator is primarily used for sample acquisition on the rover.

In addition to the above tools, a core drill is required. The drill will take samples 1m to 2m long. A rotary percussion drill requires an axial thrust of 100 lb and thus is assumed not to be held from the rover by the arm. Drilling would benefit from a stable base and the mass of the rover should be used to help the percussion motion.

There are two tool options: to be positionable by a manipulator, or to use specialized tools that do not require an manipulator. Without an arm for positioning the tool, that tool must locate itself. To an extent, the rover could be used to position a specialized tool, but the chances are good that several tools would duplicate a common positioning mechanism if there was no arm. A manipulator is desirable to acquire samples, move samples to the various processes, gimbal special sensors (similar to metal detectors) and possibly aid the rover in mobility.

The manipulator is itself a very versatile tool. By combining the arm with various acquisition tools like a gripper, the system becomes very flexible. As stated earlier, the samples determine the desired tools (except for the drill) as shown in Figure 4.

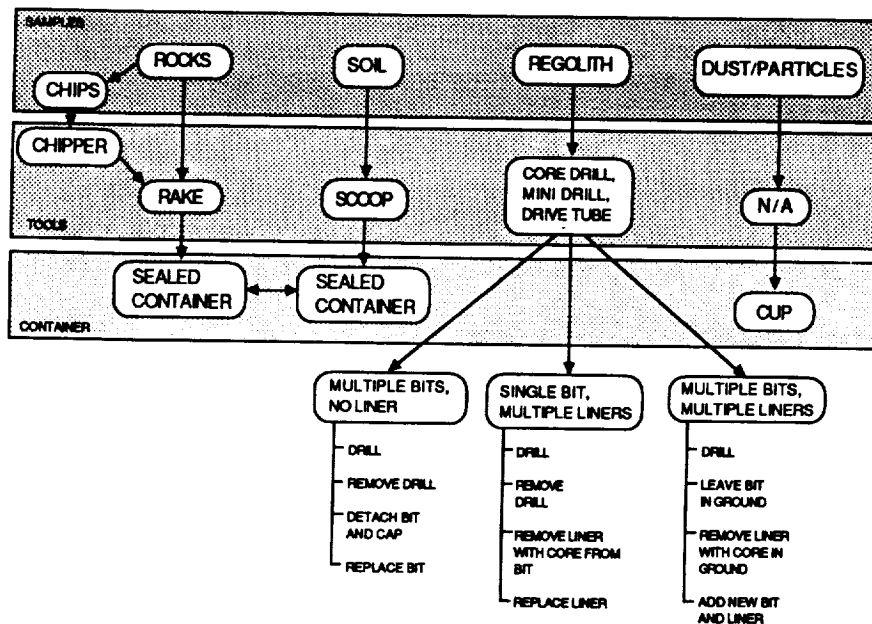


Figure 4 The Samples, Tools and Containers

The envisioned tools include a drum scoop, a chipper, a rake, a mini drill, a drive tube, and a general-purpose gripper. This list is not at all exhaustive and other tools, like a sectioning saw, a hook or additional lights and sensors are also options. The many tools require an end effector able to switch tools back and forth according to the step in the sequence.

The tools will be stored in a quick-release tool rack similar to Figure 5. In such a system, the tools will be held in an orderly manner for automated attach and detach. A compact and reliable fixture must be developed to survive launch loads and yet lock and unlock quickly and simply.

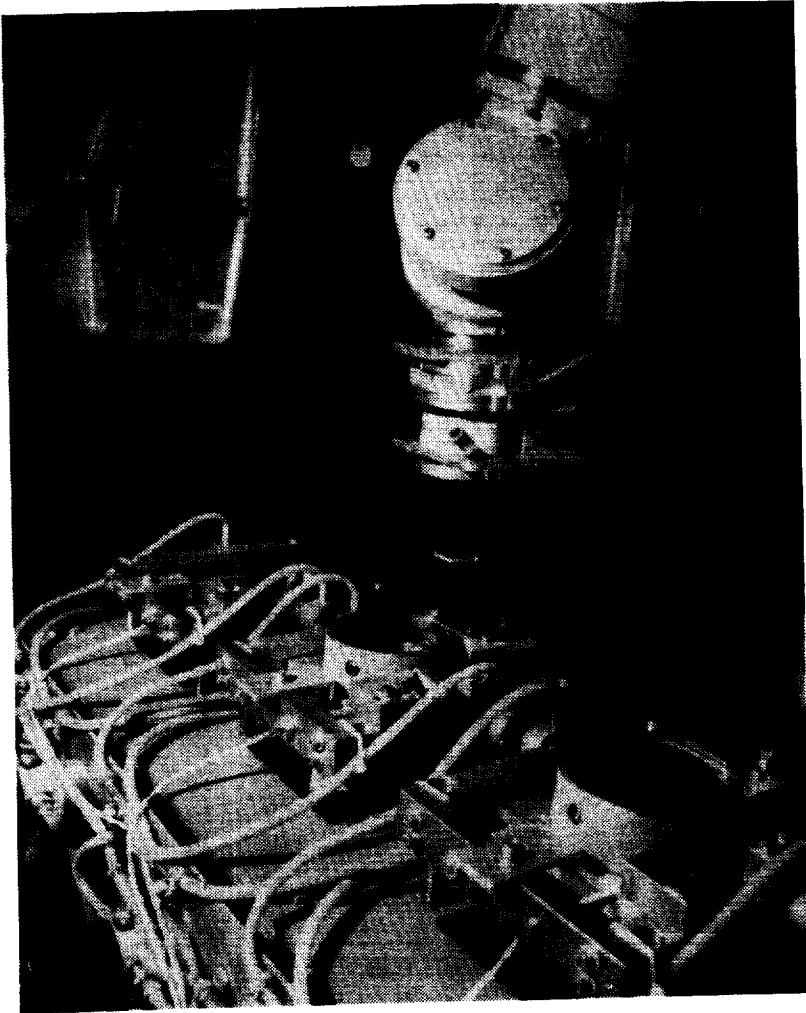


Figure 5 ITA Tool Rack

Sensors are critical to the success of this mission. The most powerful sensor is vision. Vision and possibly range imaging are needed for navigation. In addition, pictures of the environment are needed from which specimens will be picked by scientists. Another key function is the labeling of the samples.

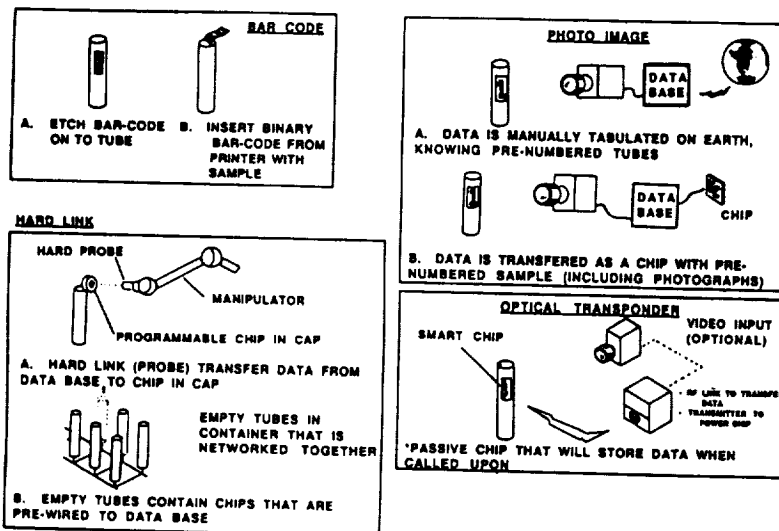


Figure 6 Labeling Options

Some examples of labeling are shown in Figure 6. The simplest method to label the individual tubes is to etch or prestamp each tube with a number or a bar code that could be read by the sampling system from all directions. It is important to identify each sample with location, temperature, preliminary certification, analysis, and local terrain conditions.

#### 4. Sampling Operations

The manipulator is a key part of the sampling system. The goal of this mission is to return 5 kg of Martian samples. It takes a year to get there, a year for exploration and a year to return for a total of a three-year mission. The first task is to scoop 200gm of bulk material with the contingency arm prior to sending out the rover.

The rover will continue in a circular traverse to acquire a diverse sampling. The anticipated 5000 gm of samples include rocks, pebbles, soils, bulk material, regolith core, and atmosphere. The rover will be navigated to a geologically interesting site. Having studied the scene, scientists on Earth would dictate the location and type of samples to test. On their command, the manipulator will go to the required tool and proceed to acquire samples. However, the system has to be flexible enough to adapt to new and different situations.

Requirements for an exploration arm have not been defined. As opposed to servicing, many of the exploration tasks are general in nature. As a result, the arm should exhibit the most capability in a certain size package. The key is versatility and is accomplished by having a flexible system. The samples to be containerized range in diameter from 2 cm to 5 cm. This does not mean the rocks will be small. "Big Joe" in the Viking photos is approximately 1 in height. A situation could arise in which the arm would have to chip away pieces from a much larger rock.

The tasks of processing, certification, analysis, containerization and preservation are well structured and controlled. These steps can be highly automated as analogous to a robotic workcell in manufacturing or assembly. The various processing, certification, analysis and containerization equipment are carefully situated for easy access. The manipulator serves to operate in

a pick-and-place condition. A robot centered cell best uses the workspace. Prior to processing, the position and orientation of the particular sample must be known. This knowledge can be obtained by vision or tactile sensing. The gripper design is general-purpose to be able to handle all samples. For an efficient workcell, the distances moved should be minimized. The cell layout is designed to reach all the stations and equipment. Considerations must be given to the movement of the end effector. Any reach aides should be minimized. Manipulator precision and accuracy is designed to meet the minimum requirements of this cell. The samples are moved sequentially from one station to another.

This manipulator is expected to perform like an assembly robot by being able to insert the samples into special tube containers. The drill also uses tubular containers for the cores. All the selected containers are gathered and transferred to the Sample Canister Assembly (SCA), which is returned to Earth.

### 5. Technical Issues

A technology cutoff date of 1992 is required for a 1998 launch. Rising costs and the long duration dictate that the mission and the technology must be reliable. Two important considerations are the time delay and the weight and power limitations on the system.

Light takes between 8 and 40 min to make the round trip. As a result, 20 percent of the available surface operations time is lost due to discontinuous communications<sup>3</sup>. Standard teleoperative control techniques would be inapplicable. Rover navigation and mobility take up additional time. Secondly, nightfall adds more restrictions in the form of downtime unless the vehicle is able to travel and work at night. High technology could help to reduce the burden.

Lighting and machine vision are important. Images of interesting sites are sent to scientists. To take full advantage of the time delay problem, they have approximately 20 min to designate an interesting area. Besides location, the type of samples desired must be input to the sampling system. The remaining sequences from acquisition to preservation should be autonomous. Periodically, scientists will have a chance to review the data of the stored samples to check their desirability. Only half the gathered samples will be returned.

Task planning algorithms will enhance the task. Force reflection will not work in this situation and intrusion detection and fault isolation will upgrade the performance of the arm. The steps from sample processing to sample preservation are hard automated (fixed). Changes in the test sequences and checks to see that the specimen will fit in the prescribed container are expected. An expert system is desirable to make decisions on the order of analysis and the overall decision to save the sample.

Greater artificial intelligence helps sampling by reducing human intervention (except for a Phobos Mission<sup>4</sup>). Understanding the capabilities of the sampling system as well as the environment is no small task. Assessing the situation is important. For example, the system may need to determine how deep a particular rock is lodged in a crater wall. From there, the arm attaches the appropriate tool and approaches the specimen from an optimized direction.

Sensor fusion (data correlation) will play an important role in this mission due to the many envisioned sensors in the system. Force sensing from manipulating an incorrect tool is just one example of versatility. There is always a chance to fail by immobilizing the arm or breaking the tool. There is a major problem if the drill jams or a drill bit snaps. Combining machine vision, force sensing, and possibly range data into a complete scene analysis will be computationally (power) intensive.

The sampling manipulator on the rover represents a design challenge. The challenge is to determine the amount of capability that can be built into a 32 kg (70.4 lb) arm, as reported in an early mass allocation. The arm will be designed for both pick and place and assembly specifications. An adjustable compliance wrist<sup>5</sup> can be applied to help satisfy both types of requirements. Packaging the arm on the rover will affect its reach and packaging in the aeroshell.

Mounted to the rover, the arm still will need a substantial reach to access everywhere. It might want to bend around a boulder and down a narrow slit to pick up a desired rock. However, the close approximations of the various sampling equipment dictate a maneuverable and dexterous manipulator. There should be no voids in the work volume. Contamination from dust storms is an issue. Bearing seals at each joint could clog or the wipers could fail. One apparent solution is to place a protective boot at each joint. The cold temperatures should minimize overheating, even if the entire structure is coated by a fine silt.

Martian gravity is .38 of Earth's gravity and consequently the 70 lb arm will be constructed from aerospace materials and advanced DC motor technology to increase performance. This could be the benchmark for a flexible manipulator. Flexible manipulators increase the load capacity of the arm without sacrificing accuracy and repeatability. Imagine the possibilities of having a long arm with the precision of an arm half its size (Fig. 7).

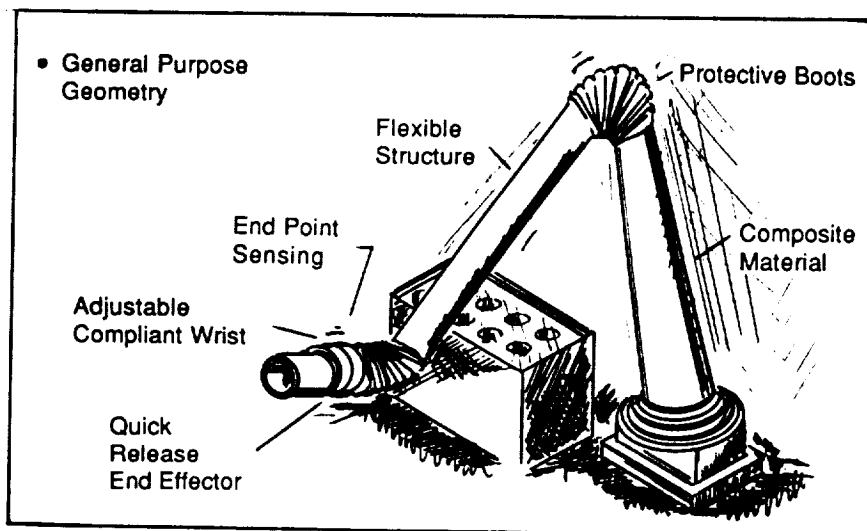


Figure 7 Anatomy of a Sampling Arm

## 6. Conclusion

Sampling operations would benefit from advanced robotic technologies. A cutoff date of 1992 requires a leap in research application and maturity. Long transmission delays dictate greater autonomy. Increased autonomy in



terms of planning, sensor fusion, expert systems and situation assessment would enhance the system.

The manipulator could benefit from a flexible structure and still be dexterous. Adjustable compliance could aid the transition from pick-and-place tasks to assembly tasks. The most important challenge is having versatility in the system without sacrificing reliability. To obtain this versatility, advanced technologies must be developed and made mature by the technology cutoff date.

The sampling mission is harder to design for than a servicing mission. Using manufacturing specifications, exploration has some general design guidelines. The technology will have to be balanced against the reliability factor. There is no room for error when the mission is three years long. The stakes are high, but the rewards promise to be greater.

#### **7. Acknowledgement**

This work was sponsored in part under, IR&D Project D-46S , Lunar and Planetary Studies.

#### **8. References**

1. H. Moore, R. Hutton, R. Scott, C. Spitzer, and R. Shorthill, "Surface Materials of the Viking Landing Sites," Journal of Geophysical Research, Vol. 82, No. 28, September 30, 1977.
2. James Gooding, "Mars Rover/Sample Return (MRSR) Mission: Description of Sample Experiments," Presentation Package at Johnson Space Center, Houston, Texas, July 13, 1988.
3. L. Allen, "Mars Rover Sample Return: Rover Challenges," AIAA/NASA First International Symposium on Space Automation and Robotics, Arlington, VA, November 29-30, 1988.
4. "Adventure 2, The Phobos Factor," U.S. News & World Report, September 26, 1988, p. 56.
5. Cutkosky, M.R., Wright, P.K.: "Active Control of a Compliant Wrist in Manufacturing Tasks." Transactions of the ASME, Journal of Engineering for Industry, February 1986, Vol. 108, pp. 36-43.



## **PARALLEL PROCESSING**



# Efficient Mapping Algorithms for Scheduling Robot Inverse Dynamics Computation on a Multiprocessor System

C. S. G. Lee

School of Electrical Engineering  
Purdue University  
West Lafayette, Indiana 47907

C. L. Chen

Department of Electrical Engineering  
Purdue University  
Indianapolis, Indiana 46205

## ABSTRACT

This paper presents two efficient mapping algorithms for scheduling the robot inverse dynamics computation consisting of  $m$  computational modules with precedence relationship to be executed on a multiprocessor system consisting of  $p$  identical homogeneous processors with processor and communication costs to achieve minimum computation time. An objective function is defined in terms of the sum of the processor finishing time and the interprocessor communication time. The minimax optimization is performed on the objective function to obtain the best mapping. This mapping problem can be formulated as a combination of the graph partitioning and the scheduling problems, both have been known to be  $NP$ -complete. Thus, to speed up the searching for a solution, two heuristic algorithms were proposed to obtain fast but suboptimal mapping solutions. The first algorithm utilizes the level and the communication intensity of the task modules to construct an ordered priority list of ready modules and the module assignment is performed by a weighted bipartite matching algorithm. For a near-optimal mapping solution, the problem can be solved by the heuristic algorithm with simulated annealing. These proposed optimization algorithms can solve various large-scale problems within a reasonable time. Computer simulations were performed to evaluate and verify the performance and the validity of the proposed mapping algorithms. Finally, experiments for computing the inverse dynamics of a six-jointed PUMA-like manipulator based on the Newton-Euler dynamic equations were implemented on an NCUBE/ten hypercube computer to verify the proposed mapping algorithms. Computer simulation and experimental results are compared and discussed.

## 1. Introduction

Robot manipulators are highly nonlinear systems, and their motion control involves the computation of the required generalized forces/torques from an appropriate manipulator dynamics model. There are a number of ways to compute the generalized forces/torques among which the computation of joint torques from the Newton-Euler (NE) equations of motion [1,2] is the most efficient and has been shown to possess the time lower bound of  $O(n)$  running in uniprocessor computers [3], where  $n$  is the number of degrees-of-freedom of the manipulator. It is unlikely that further substantial improvements in computational efficiency can be achieved. Nevertheless, some improvements could be achieved by taking advantage of particular computation structures [4], customized algorithms/architectures for specific manipulators [5,6], parallel computations [3,7], and scheduling algorithms for multiprocessor systems [8-11].

This paper presents two efficient mapping algorithms for scheduling the robot inverse dynamics computation of an  $n$ -jointed manipulator to be executed on a multiprocessor system with processor finishing time and interprocessor communication time considered in the system. The NE equations of motion are decomposed into  $m$  computational modules which are scheduled to be executed on  $p$  identical homogeneous processors to achieve minimum computation time. Several approaches to the general mapping problem have been proposed [13-18]. In this paper, an objective function is defined in terms of the sum of the processor finishing time and the interprocessor communication time. The minimax optimization of the objective function is performed to obtain the best mapping. This mapping problem is formulated as a combination of the graph partitioning and the scheduling problems; both have been known to be  $NP$ -complete. Thus, we first propose an efficient heuristic algorithm to obtain a fast but suboptimal solution. The heuristic algorithm utilizes the level and the communication intensity of the task modules to construct an ordered priority list of ready modules, and the module assignment is performed by a weighted bipartite matching algorithm. For a near-optimal mapping solution, the problem can be solved by a simulated annealing method with an efficient lower bound which indicates the minimum-finishing-time of the special scheduling problem. The simulated annealing

This work was supported in part by the National Science Foundation under Grant CDR 88-03017 to the Engineering Research Center for Intelligent Manufacturing Systems.

algorithm is a statistical optimization method which can solve various large-scale combinatorial optimization problems within a reasonable time. This approach transforms the mapping problem into a combined graph partitioning and scheduling problem. A partition of the task graph is first performed, and then the modules in each "block" of the partition are scheduled to be executed by the processor assigned to that block of the partition. Computer simulations and experimental results for computing the inverse dynamics of a six-jointed PUMA manipulator on an NCUBE/ten hypercube computer are compared and discussed.

## 2. Problem Formulation and Objective Function

The problem of computing manipulator joint torques based on a manipulator dynamic model is often referred to as the *inverse dynamics problem* and can be stated as: Given the joint positions and velocities  $\{q_j(t), \dot{q}_j(t)\}_{j=1}^n$  which describe the state of an  $n$ -jointed manipulator at time  $t$ , together with the joint accelerations  $\{\ddot{q}_j(t)\}_{j=1}^n$  which are desired at that time, solve the dynamic equations of motion for the joint torques  $\{\tau_j(t)\}_{j=1}^n$  as follow:

$$\tau(t) = f(q(t), \dot{q}(t), \ddot{q}(t)) \quad (1)$$

where  $\tau(t) = (\tau_1, \tau_2, \dots, \tau_n)^T$ ,  $q(t) = (q_1, q_2, \dots, q_n)^T$ ,  $\dot{q}(t) = (\dot{q}_1, \dot{q}_2, \dots, \dot{q}_n)^T$ ,  $\ddot{q}(t) = (\ddot{q}_1, \ddot{q}_2, \dots, \ddot{q}_n)^T$ , the superscript  $T$  denotes transpose operation on vectors and matrices, and Eq. (1) indicates the functional representation of the manipulator dynamic model. Since the NE equations of motion have been known for their efficiency in computing the joint torques, our objective is to see how fast one can map the computation of the NE equations of motion on a multiprocessor system with  $p$  identical processors to achieve minimum computation time.

### 2.1. Representation of System Model

In general, a computational task can be represented by a directed acyclic task graph (DATG)  $G_c = (V_c, E_c)$  consisting of a finite nonempty set of vertices  $V_c$ ,  $V_c = \{T_k \mid T_k \in G_c, k = 1, 2, \dots, m\}$ , and a set of finite edges,  $E_c, E_c = \{e_{ij}^c \mid e_{ij}^c \in G_c\}$ , connecting them. Each vertex represents a computational module (CM), and each edge represents the precedence constraint between two CMs. An edge connecting module  $T_i$  to module  $T_j$  is denoted by  $e_{ij}^c$ . The precedence constraint between CMs indicates which modules have to be completed before some other modules can be started. The ordered pair  $(T_i, D_i)$ † is introduced for labeling the module  $T_i$  which means that module  $T_i$  requires  $D_i$  units of execution time for completion. Similarly, a multiprocessor system can be defined by an undirected doubly weighted processor graph (UDPG)  $G_p = (V_p, E_p)$ , where  $V_p$  is a finite nonempty set of vertices denoting processors,  $|V_p| = p \neq \emptyset$ , and  $E_p = \{(e_{ij}^s, e_{ij}^r) \mid e_{ij}^s, e_{ij}^r \in G_p\}$  is a set of finite double-weighted edges.  $(e_{ij}^s, e_{ij}^r)$  is an ordered pair associated with the edge connecting processor  $i$  to processor  $j$  in  $E_p$ , where  $e_{ij}^s$  indicates the message *sending* time incurred on processor  $i$ , and  $e_{ij}^r$  indicates the message *receiving* time incurred on processor  $j$ , if the necessary message communication is required to transfer from processor  $i$  to processor  $j$ . If we represent the sending time among the processors in the system in a matrix  $SM$ , then the  $(i, j)$  entry of this matrix,  $SM(i, j)$ , indicates the sending time between the processors  $i$  and  $j$ . Similarly, the matrix  $RM$  can be used to indicate the receiving time among the processors, and the  $(i, j)$  entry of this matrix,  $RM(i, j)$ , is the receiving time between processors  $i$  and  $j$ . In this paper, we assume that these two matrices are symmetric and that the diagonal elements of these matrices are zero to indicate the negligible message communication time within the same processor. Figure 1 shows a task graph, a processor graph, and their corresponding  $SM$  and  $RM$  matrices.

For a given DATG, if there is an edge from module  $i$  to module  $j$ , then module  $i$  is said to be an immediate predecessor of module  $j$ , and we denote it as  $IPRED(j) = i$ . If there is a directed path from module  $i$  to module  $j$ , then module  $i$  is said to be a predecessor of module  $j$ , and we denote it as  $PRED(j) = i$ . Initial modules are those modules with no predecessors, and terminal modules are those modules with no successors. The level  $l_i$  of a module  $T_i$  is the summation of the execution time associated with the modules in a path from  $T_i$  to a terminal module such that this sum is maximal. Such a path is called the *critical path* if the module  $T_i$  is the highest level in the DATG [12], and we define the critical path length as

$$D_{cp} \triangleq \max_{T_i \in V_c} l_i \quad (2)$$

where  $D_{cp}$  is the minimum possible finishing time for the multiprocessors to process all the modules in the given DATG. The physical meaning of the critical path, whichever scheduling method is employed, is the finishing time over all permissible schedules and cannot be shorter than the  $D_{cp}$ .

### 2.2. Processor Finishing Time and Interprocessor Communication Time

Two important parameters are usually considered in the mapping problem: the processor finishing time (PFT) and the interprocessor communication time (PCT). The processor finishing time of a processor  $k$  for a certain

† We also alternately write  $T_i$  to represent the module  $i$ .

mapping  $S$ ,  $PFT_k(S)$ ,  $1 \leq k \leq p$ , is the accumulated execution time of the modules assigned to that processor. If there exists some precedence constraint among these modules, then the  $PFT_k(S)$  of processor  $k$  has to take the processor idle time into consideration. Determining the minimum processor finishing time with the precedence constraint taken into consideration is known as the minimum-finishing-time scheduling problem. The interprocessor communication time occurs only when two communicating modules are assigned to different processors such that the two modules residing on different processors can communicate through a communication channel or bus [15]. The message sent from one processor to other receiving processors causes the communication overhead among the sending and receiving processors. This results in the PCT which requires processing load on both the sending and receiving processors. For example, a specific module is sending a message from processor  $i$  to processor  $j$ . The sending processor  $i$  needs to spend time on message formatting and addresses initialization to the destination. Meanwhile, the receiving processor  $j$  needs to spend time on extracting the message contents from the sending processor. The message extracting time is usually much longer than the sending time which results in heavy traffic in the communication channel and causes the blocking of further messages arriving into the receiving processor  $j$ . Such PCT overhead can be eliminated if the two communicating modules are assigned to the same processor because both modules would locate at the same local memory, and the message communication time within the same processor is negligible and can be ignored.

### 2.3. Objective Function and Optimization Criterion

For most multiprocessor systems, minimizing the maximum processor finishing time is the most important performance measure and has the effect of evenly distributing the modules to all the processors to achieve the shortest possible computation time. In a task graph with the precedence constraint imposed on the modules, satisfying this mapping performance is known as the minimum-finishing-time scheduling problem. However, this performance criterion causes a negative effect which results in heavy communication costs among the processors (i.e., PCT). Minimizing the PCT alone may not produce a good assignment either because in a homogeneous system where all the processors are identical, a minimum PCT cost assignment will assign all the modules to a single processor, thus achieving zero PCT! The conflict of mapping performance between PFT and PCT has been studied [13].

The mapping problem is to find an optimal matching between a task graph and a processor graph, and this problem can be considered as a partition of the given DATG into several blocks with the modules in each block assigned to be executed by a corresponding processor. Figure 2 illustrates a partition of a DATG into three blocks to be executed on a three-processor computer system. Efficiency of a mapping can be improved by balancing the task modules among all the processors and minimizing the weight of the edges across the boundaries of the blocks which correspond to the communication costs among the processors.

Let  $S$  be a certain mapping and  $\Omega$  be the set of all the mappings. Let  $\phi_k$ ,  $1 \leq k \leq p$ , be a partition of the given DATG such that  $\bigcup_{k=1}^p \phi_k = V_c$  and  $\phi_h \cap \phi_k = \emptyset$ ,  $h \neq k$ . The PCT incurred by processor  $k$  in the  $k$ th block,  $\phi_k$ , is given by

$$PCT_k(S) = \sum_{i \in \phi_k, j, l \notin \phi_k} e_{ij}^e e_{ij}^s + e_{il}^e e_{il}^r \quad (3)$$

where modules  $i$ ,  $j$ , and  $k$  are related by  $IPRED(j) = i$  and  $IPRED(i) = l$ , and

$$e_{ij}^e = \begin{cases} 0, & \text{if } IPRED(j) \neq i \\ 1, & \text{if } IPRED(j) = i \end{cases} \quad (4)$$

Let  $E_k(S)$ ,  $1 \leq k \leq p$ , be the processor response time spent in processor  $k$  which consists of the sum of the processor finishing time  $PFT_k(S)$  and the interprocessor communication time  $PCT_k(S)$ ,

$$E_k(S) = PFT_k(S) + PCT_k(S), \quad (5)$$

where the  $PCT_k(S)$  is defined as in Eq. (3), and the  $PFT_k(S)$  is the accumulated execution time of the modules assigned to that processor. The  $PFT_k(S)$  is not simply the summation of the modules' execution time assigned to processor  $k$ ; instead, it must include the processor idle time of that processor due to the precedence constraint. Usually, this value is greater than the summation of the modules' execution time assigned to that processor. If the system is initialized, the  $PFT_k(S)$  is defined as the finishing time of executing the last module assigned to processor  $k$ .

Let  $E(S)$  be the maximum processor response time. For a mapping  $S$ ,  $S \in \Omega$ , the minimum-response-time mapping is the mapping  $S^*$  that minimizes the  $E(S)$ , that is,

$$E(S^*) = \min_{S \in \Omega} E(S) = \min_{S \in \Omega} \max_{1 \leq k \leq p} (PFT_k(S) + PCT_k(S)). \quad (6)$$

This means that we want to minimize the maximum processor response time, resulting in the minimax optimization criterion.

## 2.4. Task Graph of Newton-Euler Equations of Motion

To achieve parallel processing with minimum response time, it is desirable to develop a directed task graph with maximum parallelism for the NE equations of motion. Unfortunately, a maximum-parallelism NE task graph may not yield a minimum-response-time mapping when the interprocessor communication time is taken into consideration. Thus, we perform a functional decomposition of the NE equations; that is, the equations are decomposed into computational modules, each of which calculates the kinematic and dynamic variables such as angular velocities, angular and linear accelerations, joint forces and moments, etc. The recursive structure of NE formulation with respect to the link coordinate systems is found to be in an inhomogeneous linear recurrence form (IHLR) which is not efficient for parallel processing [3,11]. On the other hand, when expressed in the base coordinate system, the NE equations are in a homogeneous linear recurrence form (HLR) which is more suitable for parallel processing [3,11]. The NE equations of motion expressed as HLR form with the detailed task modules for our mapping problem are shown in Fig. 3. For a six-jointed, PUMA-like manipulator, this task graph shows that the NE equations can be decomposed into 145 task modules [24].

Using the minimax optimization criterion as in Eq. (6), the mapping between the above NE task graph and a set of connected processors will be investigated. Our proposed mapping strategies for a multiprocessor system are substantially different from previous work, and their contributions can be seen in the following: (1) the precedence constraint and unequal module execution time of the task modules in the task graph are considered in the system; (2) unequal communication costs among processors in the processor graph are considered in the objective function; and (3) the minimax optimization criterion is used to obtain the minimum-response-time mapping. The proposed mapping algorithms are very general and can be applied to most multiprocessor systems.

## 3. The Mapping Algorithm

The mapping problem can be considered as decomposing a computational task into a set of task modules executed concurrently on a multiprocessor system. The design of this mapping is divided into two major steps: graph partitioning and module allocation. Both of these two steps are known to be *NP*-complete. Thus, we propose two heuristic mapping algorithms to obtain fast mapping solutions for computing the NE equations of motion. Both heuristic algorithms take the PFT, the PCT, the precedence constraint of the NE task graph, and the multiprocessor system structure into consideration.

### 3.1. Heuristic Mapping Algorithm with Weighted Bipartite Matching

Based on the given NE task graph in Fig. 3, the heuristic algorithm constructs a *dynamic* priority list containing all the task modules arranged in a descending order according to the weighted level  $l_i$  and the communication intensity of the task modules. Most of the priority lists developed in previous research do not include the communication costs [9,11]. To develop the dynamic priority list, let us denote  $A(n)$  to be a set of modules that have been assigned to the processors at the  $n$ th insertion stage (i.e., the modules that have been inserted into the mapping-schedule from the dynamic priority list), and let  $\bar{A}(n)$  be the complement of  $A(n)$ . Let  $P_{mfi}(n)$  be the processor(s) with the minimum finishing time at this stage, and let  $K(n)$  denote the set of modules assigned to the remaining processors which have not yet finished processing. Let  $FW(\bar{A}(n))$  be the function that returns the set of modules,  $W(n)$ , which are ready to be assigned to all the  $p$  processors, that is, for all modules  $T_i \in \bar{A}(n)$ , if and only if  $PRED(T_i) \in \bar{A}(n)$ . Similarly, the function  $FW(K(n) \cup \bar{A}(n)) - K(n)$  returns the set of modules,  $R(n)$ , which are ready to be assigned to the  $P_{mfi}(n)$ . These notations will be useful for developing the proposed heuristic algorithm.

The mapping-schedule obtained from this heuristic algorithm starts from zero initially and gradually the ready modules are "inserted" into the mapping-schedule until all the modules have been inserted. Instead of randomly inserting the ready modules into the available processors  $P_{mfi}(n)$ , the insertion of ready modules needs to consider the increased communication costs they created. In order to minimize the maximum processor response time, this insertion of ready modules has to be carefully selected so that it does not increase the PCT while maintaining the minimum-finishing-time schedule. This insertion process (or module allocation process) can be done by a weighted *Bipartite Matching Algorithm* [20] which will be discussed later. Using the weighted bipartite matching algorithm at each stage of the insertion of ready modules into the mapping-schedule, the PCT will be maintained to be the minimum.

Minimizing the maximum processor response time also requires us to consider the PFT at each stage of the insertion of ready modules. Due to the precedence constraint of the task graph, some idle modules might have to be assigned to the available processors during the insertion of the ready modules. Table 1 shows the PFT of the mapping-schedule in Fig. 1 when the new non-idle module is inserted to that processor. In order to include both the PFT and the PCT in the construction of the dynamic priority list to order the ready modules, we introduce two parameters,  $\alpha$  and  $\beta$ , for weighting the level  $l_i$  and the communication intensity of task modules, respectively, in adjusting the assignment priority coefficient  $p_i$ . The ready modules are ordered into a priority list according to the descending order of the assignment priority coefficient  $p_i$ . Then the ready modules in the priority list can be allocated to the available



processors using the weighted bipartite matching algorithm.

**Algorithm HM (Heuristic Mapping Algorithm).** Given a DATG and an UDPG, this algorithm constructs a dynamic priority list of all the task modules and inserts the modules one by one into the mapping-schedule according to the weighted bipartite matching algorithm.

BBH1.[Initialization.] Input the given DATG. Obtain the critical path  $D_{cp}$  and the total number of edges,  $nedge$ , in the DATG, where  $nedge = |E_c|$ .

H2. [Initialize Looping.] Set the mapping-schedule empty (i.e.,  $A(n) = \emptyset$ ); initialize the processor finishing time PFT and the interprocessor communication time PCT. Create two parameters  $\alpha$  and  $\beta$  ( $\alpha + \beta = 1$ ),  $\alpha$  from one to zero and  $\beta$  from zero to one with an increment/decrement  $\Delta STEP$ . Set  $n \leftarrow 1$ .

H3. [Determine the level of modules in  $R(n)$ .] Determine the set of ready modules  $R(n)$ , and find the level  $l_i$  of each module  $T_i$  in the set  $R(n)$ .

H4. [Obtain the successors and the predecessors number.] For each module  $T_i$  in the set  $R(n)$ , find its number of successors,  $ns_i$ , and its number of predecessors,  $np_i$ .

H5. [Evaluate the assignment priority coefficient.] Evaluate the assignment priority coefficient,  $\rho_i$ , of module  $T_i$  from the equation

$$\rho_i = \alpha \left( \frac{l_i}{D_{cp}} \right) + \beta \left( \frac{ns_i + np_i}{nedge} \right), \text{ where } \alpha + \beta = 1. \quad (7)$$

H6. [Order the priority list.] For all the ready modules in the set  $R(n)$ , construct an ordered priority list  $R_l(n)$  according to the descending order of  $\rho_i$ .

H7. [Assign the modules.] Determine the number of available processors  $p_{av} = |P_{mft}(n)|$ . According to the weighted bipartite matching algorithm, assign the first  $p_{av}$  modules to the available processors based on the priority list  $R_l(n)$ . Record the modules in each processor, the PFT, and the PCT.

H8. [End of mapping-schedule?] If  $\bar{A}(n) \neq \emptyset$ , then set  $n \leftarrow n + 1$ , and go to step H3; otherwise, continue.

H9. [Obtain minimum cost among all mappings.] If  $\bar{A}(n) = \emptyset$ , record the mapping-schedule and the costs of this mapping. If  $\alpha \neq 1$ , then go to step H2 to obtain another mapping; otherwise, stop and obtain the minimum cost among all the mappings.

#### END HM.

In the above HM algorithm, based on the level  $l_i$  and the communication intensity of the task modules in the DATG, the weighting parameters,  $\alpha$  and  $\beta$ , are used to evaluate the assignment priority coefficient  $\rho_i$  which is used to order the priority list  $R_l(n)$  (steps H5 and H6). The parameters  $\alpha$  and  $\beta$  are used to weight the level and the communication intensity of the task modules, respectively. In the extreme case, when  $\alpha = 1$ , the ready modules are ordered according to their level in the task graph. Similarly, when  $\beta = 1$ , the ready modules are ordered according to the communication intensity of the task modules in the task graph. By suitably adjusting  $\alpha$  and  $\beta$ , various orderings of the ready modules in the priority list at each stage of the insertion can be generated, thus providing a better mapping solution.

The weighted bipartite matching problem is also known as the "job-worker" assignment problem. Each worker must be assigned to exactly one job, and each job must have one assigned worker. If job  $i$  is assigned to worker  $j$ , then a benefit of the objective function is realized. It is desirable to make an assignment such that the total benefit of the objective function is optimized. In our mapping problem, the jobs and workers are, respectively, corresponding to the ready modules and the available processors at each stage of the insertion. Since there are  $p_{av}$  available processors at each stage of the insertion, the *weights* or *benefits* produced by job  $i$  (or ready module  $i$ ) assigned to worker  $j$  (or available processor  $j$ ) is not simply a scalar value. Instead, the weights are expressed in a  $p_{av}$ -tuple, each of the  $k$ th element in this  $p_{av}$ -tuple is the benefit obtained by the  $k$ th processor in this assignment. We want to minimize the maximum element of this  $p_{av}$ -tuple to obtain the minimum cost of the assignment. Before we formulate the weighted bipartite matching algorithm, two operations are defined on the ordered  $p$ -tuples. Let  $D = (d_1, d_2, \dots, d_p)$  be an ordered  $p$ -tuple and  $U = (u_1, u_2, \dots, u_p)$  be another ordered  $p$ -tuple. Then, the sum operation of these two ordered  $p$ -tuples, written as  $D \oplus U$ , results in another  $p$ -tuple,  $W$ ,

$$W = (w_1, w_2, \dots, w_p) = D \oplus U \triangleq (d_1 + u_1, d_2 + u_2, \dots, d_p + u_p). \quad (8)$$

The maximum value of this  $p$ -tuple, written as  $\max_p$ , is the maximum element in  $W$ , that is,  $\max_p \triangleq \max_{1 \leq i \leq p} w_i$ . The summation of a series of  $p$ -tuples is written as  $\sum_{\oplus}$ .

**Algorithm WBM (Weighted Bipartite Matching Algorithm).** Given an ordered priority list of the ready modules  $R_i(n)$ , the available processors, the number of available processors, the mapping-schedule at the  $n$ th stage, the first  $p_{av}$  ready modules denoted as  $R_i^{P_n}(n)$ , the matrix  $RM$ , the  $p_{av}$ -tuple processor finishing time  $PFT^{P_n}(n-1)$ , and the  $p_{av}$ -tuple interprocessor communication time  $PCT^{P_n}(n-1)$ , this algorithm assigns the ready modules to the available processors such that the PCT among all the processors is minimum.

W1. [Obtain communication time vector.] For module  $y$  in  $R_i^{P_n}(n)$ , obtain the number of its predecessors,  $n_{yk}$ , located in processor  $k$ ,  $1 \leq k \leq p_{av}$ . Construct a  $p_{av} \times 1$  vector; each entry of this vector is a  $p_{av}$ -tuple. The  $k$ th element of the  $i$ th entry of this vector,  $k \neq i$ , is the PCT incurred on processor  $k$  if module  $y$  is assigned to processor  $i$  and is calculated as  $\sigma_k = RM(i, k)n_{yk}$ . The  $i$ th element of the  $i$ th entry of the communication time vector is  $\sum_{k=1, k \neq i}^{p_{av}} \sigma_k$ . This is due to the PCT incurred on processor  $i$  and is defined as the sum of the weights of the edges across the boundary of the block  $i$  as we indicated in Eq. (3).

W2. [Obtain communication time matrix.] For all the ready modules in  $R_i^{P_n}(n)$ , construct the  $p_{av} \times p_{av}$  communication time matrix  $\Gamma(n)$  by collecting all the communication time vectors. Note that the  $i$ th column of this matrix corresponds to the communication time vector which is created by the module  $y$ , and this module is located at the  $i$ th position of the  $R_i^{P_n}(n)$  priority list.

W3. [Obtain cost matrix.] Construct a  $p_{av} \times p_{av}$  cost matrix  $C(n) = [c_{ij}(n)]$  by

$$c_{ij}(n) = \Gamma_{ij}(n) \oplus (0, \dots, PFT_i^{P_n}(n-1), \dots, 0) \oplus PCT^{P_n}(n-1) \oplus (0, \dots, D_y, \dots, 0), \quad (9)$$

where  $i, j = 1, 2, \dots, p_{av}$ ,  $PFT_i^{P_n}(n-1)$  is the  $i$ th element of the  $p_{av}$ -tuple at the  $(n-1)$ th stage,  $D_y$  is the execution time of module  $y$  and is located at the  $i$ th position of the  $p_{av}$ -tuple. Note that  $c_{ij}$  is a  $p_{av}$ -tuple.

W4. [Weighted bipartite assignment problem.] Solve the module assignment problem by finding an assignment matrix  $X(n) = [x_{ij}(n)]$ ,  $i, j = 1, 2, \dots, p_{av}$  to

$$\min_{X(n)} (\max_{p_{av}} \sum_i \sum_j c_{ij}(n) x_{ij}(n)) \quad (10)$$

subject to  $\sum_{j=1}^{p_{av}} x_{ij}(n) = 1$  for  $i = 1, 2, \dots, p_{av}$ ,  $\sum_{i=1}^{p_{av}} x_{ij}(n) = 1$  for  $j = 1, 2, \dots, p_{av}$ , and  $x_{ij}(n) = 1$  or  $0$ .

W5. [Assign modules to processors.] According to the module assignment matrix  $X(n)$ , assign the modules in  $R_i^{P_n}(n)$  to the available processors.

**END WBM.**

When inserting the  $p_{av}$  ready modules into the mapping-schedule at the  $n$ th stage, the cost matrix  $C=[c_{ij}(n)]$  in Eq. (9) includes the communication time the modules are going to create with the assigned modules in the  $A(n)$ , the processor finishing time of the mapping-schedule,  $PFT^{P_n}(n-1)$ , the interprocessor communication time of the mapping-schedule,  $PCT^{P_n}(n-1)$ , and the execution time of the modules to guarantee the best assignment.

As an example, consider the mapping-schedule of the given DATG in Fig. 1 ( $m=9$ ). The task modules are to be executed by 3 processors ( $p=3$ ). The level number and execution time of each module are given beside each module in the DATG, the  $e_{ij}^l$  and  $e_{ij}^r$ ,  $i, j = 1, 2, 3$ , are shown in the UDPG. We use the HM algorithm and let  $\alpha=0$ ,  $\beta=1$  to determine a heuristic mapping-schedule. The  $R(n)$ ,  $p_{av}$ ,  $R_i^{P_n}(n)$ ,  $PFT(n)$ , and  $PCT(n)$  associated with each stage of insertion are listed in Table 2. The PFT and PCT of this mapping are found to be  $PFT = (13, 15, 7)$  and  $PCT = (6, 7, 7)$ ; the cost of this mapping is 22 units, and the Gantt chart of this suboptimal mapping is shown in Fig. 4.

### 3.2. Heuristic Mapping Algorithm with Simulated Annealing

In order to improve the mapping solution obtained by the above HM algorithm, the simulated annealing method [21], which incorporates an iterative improvement scheme and the Metropolis procedure, is introduced to find a near-optimal mapping. Consider that one starts with an initial state (i.e., a partition of the task graph DATG) of the optimization problem, instead of always rejecting the new state that increases the objective function, this new state with small conditional probability is accepted. In other words, if  $\Delta E \leq 0$ , then accept this new state; if  $\Delta E > 0$ , then accept this new state with probability  $e^{-\Delta E/T}$ , where  $\Delta E = E(S_{new}) - E(S)$  is the amount of the increase in the maximum processor response time  $E(S)$ ,  $S_{new}$  and  $S$  are, respectively, the new and old states, and  $T$  is a control parameter. Initially, one starts with a large value of  $T$ ; after the system is approaching the equilibrium at this  $T$  value, then  $T$  is reduced to a lower value and the system will approach to another equilibrium. This procedure is stopped at a certain desirable  $T$ , or no more improvement can be expected. By conditionally accepting the increased  $E(S_{new})$  and by varying the control parameter  $T$ , one can escape the pitfall of the local optimal and obtain a better mapping.

Applying this simulated annealing method to our mapping problem, a random state is generated when all the modules in the DATG are randomly assigned to the processors and the PCT of this assignment is obtained (this is the graph partitioning problem); then based on the modules assigned to these processors, we schedule the computation of these modules in the processors according to the precedence constraint imposed on them and the PFT of this assignment is obtained (this is the scheduling problem). To compute the objective function of a new state, one needs to compute both the PCT and the PFT. The PCT of a state is simply the weights of all the edges across the boundaries of each block while the PFT of this state is difficult to obtain because obtaining the optimal schedule is difficult. Thus, one may use the heuristic scheduling algorithm [11] to obtain a suboptimal schedule or derive the lower bound of the PFT of the schedule. Since the graph partitioning randomly assigns task modules to be executed on specific processors, the scheduling problem that follows needs to address the restriction of executing these modules on their specific processors. This scheduling problem can be solved by the proposed processor-restricted dynamical-highest-level-first/most-immediate-successors-first algorithm (Algorithm PRDHLF/MISF).

**Algorithm PRDHLF/MISF.** (*Processor-Restricted Dynamical Highest Level First/Most Immediate Successors First Algorithm*). Given a task graph and restricting the execution of the modules on specific processors, this algorithm constructs a dynamic priority list of all the modules and inserts the modules one by one into the suboptimal schedule.

- D1. [Initialization.] Initially, the schedule is empty (i.e.  $A(n) = \emptyset$ ). Let  $P_i$  be the set of modules that are pre-assigned to the processor  $i$ ,  $i = 1, 2, \dots, p$ . Let  $P_{mfi}^i(n)$ ,  $i = 1, 2, \dots, p$ , be the processor  $i$  having the minimum-finishing-time at the  $n$ th stage of the insertion.
- D2. [Determine the set  $R(n)$  and the sets  $R_i(n)$ .] Determine the set  $R(n)$  and obtain  $R_i(n) = R(n) \cap P_i$ ,  $i = 1, 2, \dots, p$ , where  $R_i(n)$  is the set of ready modules assigned to the processor  $i$ .
- D3. [Determine the levels of modules in  $R_i(n)$ .] Find the level  $l_k$ ,  $T_k \in R_i(n)$ , for each module in the set  $R_i(n)$ ,  $i = 1, 2, \dots, p$ .
- D4. [Construct the priority lists.] Construct the  $i$ th dynamic priority list of processor  $i$  in a descending order of  $l_k$ . If the levels of the modules are tied, then the module having the largest number of immediately successive modules is assigned to the highest priority.
- D5. [Assign the modules.] Assign the modules to the  $P_{mfi}^i(n)$  on the basis of the  $i$ th priority list,  $i = 1, 2, \dots, p$ . If  $\bar{A}(n) = \emptyset$ , then stop; otherwise, go to step D2.

**END PRDHLF/MISF.**

Although the PRDHLF/MISF algorithm provides a fast but suboptimal schedule, it is still very time-consuming to obtain the schedule when the number of task modules is large. Since our objective is to compute the PFT instead of finding a complete schedule, it is more desirable to find the lower bound of the PFT of the schedule. The lower bound of the PFT of the schedule without the restriction of executing the modules on specific processors has been discussed [22]. The lower bound of the PFT of the schedule with the restriction of executing the modules on specific processors is given by

$$PFT_i^{LB}(S) = \max_{T_j \in P_i} [\tau_i^H(T_j) + q_i] \quad (11)$$

where the notation and the proof of this lower bound can be found in [24].

With the lower bound of the PFT in Eq. (11) and the PCT obtained from the graph partitioning, the objective function of a state  $S$  can be easily obtained by

$$E(S) = \max_{1 \leq k \leq p} (PFT_k^{LB}(S) + PCT_k(S)). \quad (12)$$

Using the objective function in Eq. (12), the heuristic mapping algorithm with simulated annealing to obtain a near-optimal mapping is summarized in the following.

**Algorithm HMSA** (*Heuristic Mapping Algorithm with Simulated Annealing*) Given a DATG, an UDPG, the matrices  $SM$  and  $RM$ , and the control parameter  $T$ , this algorithm generates a near-optimal solution of the mapping problem.

- S1. [Initialization.] Select an initial state  $S$ , and evaluate the objective function  $E(S)$  in Eq. (12).
- S2. [Looping with  $T$ .] If the equilibrium with respect to the control parameter  $T$  is reached, go to step S6; otherwise continue.
- S3. [Generate new state.] Generate a new state  $S_{new}$ , and calculate  $E(S_{new})$  of this new state according to Eq. (12). Obtain  $\Delta E = E(S_{new}) - E(S)$ . Select a random variable  $\gamma$ ,  $0 \leq \gamma \leq 1$ . If  $\Delta E \leq 0$ , go to step S5; otherwise go to step S4.

- S4. [ $\Delta E > 0$ .] If  $\gamma < e^{-\Delta E/T}$ , then accept this new state with probability  $e^{-\Delta E/T}$  and go to step S5; otherwise reject this new state and go to step S3.
- S5. [ $\Delta E \leq 0$  or  $\gamma < e^{-\Delta E/T}$ .] Set  $S \leftarrow S_{new}$ . Record  $E(S_{new})$  and go to step S2.
- S6. [Looping with a smaller value of  $T$ .] If  $T > \text{stopping criterion}$ , set  $T \leftarrow T - \Delta T$ , where  $\Delta T$  is a user-designed variable, and go to step S3; otherwise, continue.
- S7. [Obtain the mapping.] Obtain the state which has the smallest  $E(S)$ . Based on this state, schedule the modules according to the precedence constraint of the DATG and obtain the mapping.

END HMSA.

The HMSA algorithm is used to find a near-optimal mapping for the example in Fig. 1. An initial partition of the task graph assigns the modules  $\{T_1, T_2, T_3\}$ ,  $\{T_7, T_8, T_9\}$ , and  $\{T_4, T_5, T_6\}$  to processors 1, 2, and 3 for execution, respectively. At this initial state, the  $PFT_i^{LB}$  for  $i = 1, 2, 3$  are respectively 10, 21, 15, and the  $PCT_i$  are 5, 12, and 21. The total cost of this state is 36 units. In this example, 273 iterations were required to reach the near-optimal mapping. The  $PFT_i^{LB}$  for the final state are 12, 15, and 13, and the  $PCT_i$  are 8, 7, and 9. The total cost of this final state is 22 units, and the partition of the task graph for this mapping assigns the modules  $\{T_3, T_6, T_7\}$ ,  $\{T_5, T_9\}$ , and  $\{T_1, T_2, T_4, T_8\}$  to processors 1, 2, and 3 for execution, respectively. It is interesting to note that both the HM and HMSA algorithms reached the same total cost of 22 units with different mapping solutions. Further computer simulations and actual implementation of the proposed algorithms on an NCUBE/ten multiprocessor system are described in the next section.

#### 4. Computer Simulations and Experimental Results

Since the optimal mapping solution is *NP*-complete, the design of computer simulations for evaluating the efficiency of the proposed HM and HMSA algorithms as compared with the optimal solution is constrained by the number of task modules and processors that our computer can process within a reasonable time. A total of 100 random DATGs, each with the number of modules ranging from 5 to 10, was generated for mapping onto multiprocessor systems with 2 to 3 processors. The ratio of average module execution time of the modules in the DATG and the average interprocessor communication time between paired processors,  $P/C$ , was introduced to determine the communication overhead affecting the performance of the algorithms. In our computer simulations, all the optimal solutions were obtained by the exhaustive search method. Table 3 shows the efficiency of the HM algorithm with a  $P/C$  ratio ranging from 10 to 0.1 together with different numbers of processors. To evaluate the efficiency of the heuristic algorithms, an average relative error  $\epsilon$  over  $N$  mappings is defined as

$$\epsilon = \frac{1}{N} \sum_{i=1}^N \frac{h_i - o_i^*}{o_i^*} \quad (13)$$

where  $h_i$  is the  $i$ th heuristic solution and  $o_i^*$  is the  $i$ th optimal solution.

From the computer simulation, when the  $P/C$  ratio is less than 1, the solutions obtained by the HM algorithm were found to be unsatisfactory. This is due to the fact that the HM algorithm always distributes the ready modules to the available processors to achieve the minimum communication costs. When the average communication time is much greater than the average module execution time or the communication link density is higher, evenly distributing the ready modules to all the available processors will not produce a better mapping. Instead, it may happen that all the ready modules may be assigned to the same processor to reduce the large average communication time and sacrifice relatively small average module execution time to yield a better response. Thus, when the  $P/C$  ratio is less than one, a sequential computation of all the task modules on a single processor will yield a better result than parallel processing. Hence the performance of a multiprocessor system is closely related to the  $P/C$  ratio. To improve the performance of a multiprocessor system, it is suggested that the computational task should be decomposed such that the  $P/C$  ratio is greater than one. In our computer simulation, when the  $P/C$  ratio is greater than one, all the solutions obtained by the HM algorithm approached to the near-optimal solutions. The mapping solutions obtained from the HMSA algorithm for all the 100 DATGs agreed with the optimal solutions obtained by the exhaustive search method.

To further evaluate and validate the effectiveness of our proposed mapping algorithms, the computation of the robot inverse dynamics based on the NE task graph in Fig. 3 is implemented on an NCUBE/ten multiprocessor system. Once the sending and receiving processors are given, the path is obtained by the operating system of the machine. The performance of communication activities between two processors in the hypercube computer has been addressed [23]. It is shown that the hypercube machine has a high communication cost compared with the processing time of computing elementary multiplication or addition operation. For transferring a 12-byte (or 96-bit)<sup>†</sup> message, the ratio

<sup>†</sup> For a functional decomposition, each entity of the computation is  $3 \times 1$  vector (e.g.,  $\omega_i$  etc.) and each scale value requires a 4-byte (32-bit) computation; thus the standard size to transfer a vector is 12 bytes.

of multiplication (or addition) operation and the message passing between two adjacent processors is approximately  $P/C \approx 50/510 \ll 1$ . This indicates that for implementation on NCUBE machines, it is not advisable to decompose the NE equations of motion into elementary operations. Thus, we perform functional decomposition on the NE equations of motion which results in the task graph in Fig. 3. The communication time between two processors for our NCUBE/ten machine has been measured experimentally and is found to be 0.34 milliseconds for sending a 12-byte message from a source processor to any destination processor. For receiving the message, we found that it takes 1.02, 1.7, and 2.2 milliseconds to receive a 12-byte message between one, two, and three hops<sup>‡</sup>, respectively.

A recent result indicates that it takes 24.8 milliseconds to compute the robot inverse dynamics on a uniprocessor system [8]. Due to the high communication time, if we implement the robot inverse dynamics computation on the hypercube computer, it may not generate a faster result than that running on a uniprocessor computer. In order to avoid mapping all the modules to a single processor by the proposed heuristic algorithms, we artificially set the execution time of a 12-byte elementary operation (multiplication or addition) to 3.4 milliseconds which is ten times of the time for sending a 12-byte message. Based on the NE task graph in Fig. 3, for a 6-jointed PUMA-like manipulator, the NE equations can be decomposed into 145 task modules. The detailed computation time and precedence relationship of each module can be found in [24]. Using the sending time matrix  $SM$  and the receiving time matrix  $RM$  of a subcube of four processors of our NCUBE/ten computer, we simulated on a VAX-11/780 computer the HM algorithm for finding our mapping solution. Based on the 145-module NE task graph, a suboptimal mapping was obtained from this computer simulation. The minimum processor response time of each processor from the suboptimal mapping was found to be 704.82, 701.42, 706.52, and 703.12 milliseconds for processor 0, 1, 2, and 3, respectively. We then implemented the mapping solution from the computer simulation into the actual NCUBE/ten machine. The minimum processor response time of each processor from the mapping was found to be 679.71, 676.69, 675.78, and 669.23 milliseconds for processor 0, 1, 2, and 3, respectively. Table 4 compares the computer simulation result with the actual implementation result and shows a maximum of five percent relative error between them.

In order to achieve the critical-path-length computation of the 145-module NE task graph, the hypercube dimension was increased from  $N = 2$  to  $N = 3$ ; that is, 8 processors were used for the computations. The details of the allocation of the 145 modules in each of the eight processors can be found in [24]. Using  $p = 8$  processors, our HM algorithm achieves the critical-path-length computation of 400.18 milliseconds which incurred in processor number 3; with a communication time of 27.86 milliseconds, it gave a shortest possible response time of 428.04 milliseconds. The critical-path-length computation of 400.18 milliseconds means that the use of more than 8 processors for parallel processing will never obtain a shorter processing time. The comparison of the computer simulation result and the implementation result is shown in Table 5.

## 5. Conclusions

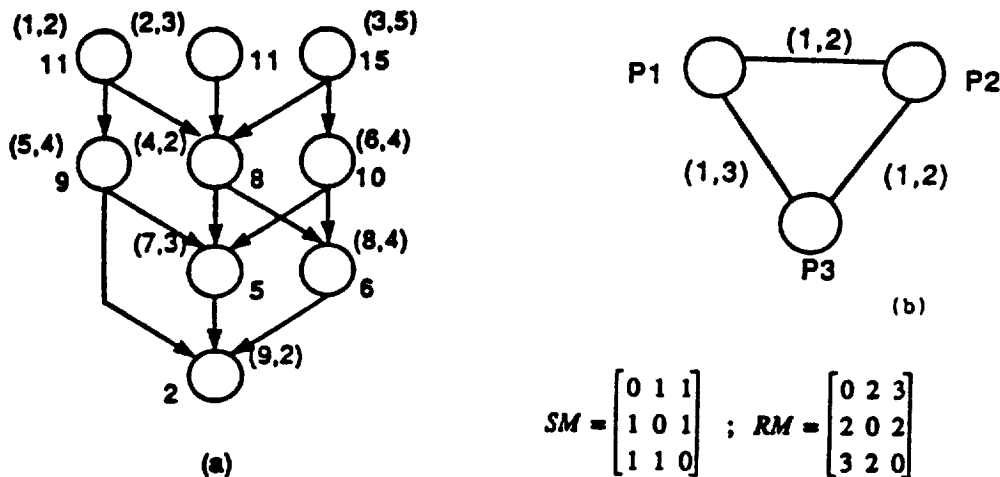
The problem of determining the optimal mapping between a computational task consisting of  $m$  modules with precedence constraint and a set of connected  $p$  identical processors with interprocessor communication time is formulated as an equivalent problem of solving the graph partitioning and the scheduling problems. Both of these problems are known to be  $NP$ -complete. Two efficient mapping algorithms, the HM algorithm and the HMSA algorithm, were proposed to determine fast and suboptimal or near-optimal mapping solutions. Minimizing the maximum of the sum of the processor finishing time and the interprocessor communication time is used as an optimization criterion for determining the best mapping. The proposed HM algorithm and the HMSA algorithm greatly reduce the time complexity of determining the mapping. Computer simulation results indicated that the proposed mapping algorithms are efficient and practical and that they can provide suboptimal as well as near-optimal solutions. Computer simulation was also conducted to simulate the operation of an NCUBE/ten hypercube computer for determining the mapping of the robot inverse dynamics computation of a 6-jointed PUMA-like manipulator. Actual implementation of the HM algorithm on the NCUBE/ten hypercube computer was also performed.

## References

- [1] J. Y. S. Luh, M. W. Walker, and R. P. Paul, "On-line Computational Scheme for Mechanical Manipulator," *Trans. ASME, J. Dynam., Syst., Meas., Contr.*, vol. 120, pp. 69-76, June 1980.
- [2] D. E. Orin, R. B. MaChee, M. Vukobratovic, and G. Hartoch, "Kinematics and Kinetic Analysis of Open-Chain Linkages Utilizing Newton-Euler Methods," *Math. Biosci.*, vol. 43, pp. 107-130, 1979.
- [3] C. S. G. Lee and P. R. Chang, "Efficient Parallel Algorithm for Robot Inverse Dynamics Computation," *IEEE Trans. Syst. Man, and Cybern.*, vol. SMC-16, no. 4, pp. 532-542, July 1986.
- [4] C. S. G. Lee, T. N. Mudge, and J. L. Turney, "Hierarchical Control Structure Using Special Purpose Processor for the Control of Robot Arm," *Proc. 1982 Conf. Patt. Recog. and Image Processing*, pp. 634-640, June 1982.

<sup>‡</sup>The hops is the number of edges of the hypercube which the message must traverse when the message is sending from the source processor to the destination processor.

- [5] R. Nigam and C. S. G. Lee, "A Multiprocessor-Based Controller for Control of Mechanical Manipulators," *IEEE J. of Robotics and Autom.*, vol. RA-1, no. 4, pp. 173-182, Dec. 1985
- [6] T. Kanade, P. K. Khosla, and N. Tanaka, "Real-Time Control of the CMU Direct Arm II Using Customized Inverse Dynamics," *Proc. of IEEE Conf. on Decision and Contr.*, pp. 1345-1352, Dec. 1984.
- [7] L. H. Lathrop, "Parallelism in Manipulator Dynamics," *Int'l J. of Robotics Res.*, vol. 4, no. 2, pp. 80-102, Summer 1985.
- [8] J. Y. S. Luh and C. S. Lin, "Scheduling of Parallel Computer for a Computer-Controlled Mechanical Manipulator," *IEEE Trans. Syst. Man, and Cybern.*, vol. 12, pp. 214-234, 1982.
- [9] H. Kasahara and S. Narita, "Parallel Processing of Robot-Arm Control Computation on a Multiprocessor System," *IEEE J. of Robotics and Autom.*, vol. RA-1, no. 2, pp. 104-113, June 1985.
- [10] J. Barhen, "Robot Inverse Dynamics on a Concurrent Computation Ensemble," *Proc. of 1985 ASME Int'l Conf. on Computers in Engineering*, vol. 3, pp. 415-429, 1985.
- [11] C. L. Chen, C. S. G. Lee, and E. S. H. Hou, "Efficient Scheduling Algorithms of Robot Inverse Dynamics Computation on a Multiprocessor System," *IEEE Trans. Syst. Man, and Cybern.*, vol. SMC-18, no. 5, September/October 1988.
- [12] E. G. Coffman, *Computer and Job-Shop Scheduling Theory*, Wiley, New York, 1976.
- [13] K. Efe, "Heuristic Models of Task Assignment Scheduling in distributed Systems," *Computer*, vol. 15 pp. 50-56, July 1982.
- [14] S. H. Bokhari, "On the Mapping Problem," *IEEE Trans. Computer*, vol. C-30, pp. 207-214, Mar. 1981.
- [15] W. W. Chu and L. M-T Lan, "Task Allocation and Precedence Relations for Distributed Real-Time Systems," *IEEE Trans. Computer*, vol. C-36, pp. 667-679, June 1987.
- [16] S. Y. Lee and J. K. Aggarwal, "A Mapping Strategy for Parallel Processing," *IEEE Trans. Computer*, vol. C-36, pp. 433-441, April 1987.
- [17] E. R. Barnes, "An Algorithm for Partitioning the Nodes of a Graph," *SIAM J. Alg. Disc. Math.*, vol. 3 no. 4, pp. 541-550, Dec. 1982.
- [18] W. E. Domath and A. J. Hoffman, "Lower Bounds for the Partitioning of Graphs," *IBM J. Res. Develop.*, vol. 17, pp. 420-425, Sept. 1973.
- [19] B. W. Kernighan and S. Lin, "An Efficient Heuristic Procedure for Partitioning Graph," *Bell System Tech. Journal*, vol. 49, no. 2, pp. 291-307, Feb. 1970.
- [20] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization, Algorithms and Complexity*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1982.
- [21] S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi, "Optimization by Simulated Annealing," *Science*, vol. 220, no. 4598, pp. 671-680, 1983.
- [22] E. B. Fernandez and B. Bussell, "Bound on the Number of Processors and Time for Multiprocessor Optimal Schedules," *IEEE Trans. Computer*,
- [23] A. Beguelin and D. J. Vasicek, "Communication between Nodes of a Hypercube," *Proc. 2nd Conf. Hypercube Multiprocessors*, Knoxville, TN, pp. 162-168, Sept. 1986.
- [24] C. L. Chen, "Efficient Mapping Algorithms for Scheduling Autonomous Vehicles and Robotic Computations," Ph.D. dissertation, School of Electrical Engineering, Purdue University, August 1988.



**Figure 1.** (a) A direct task graph.  
(b) An undirected processor graph with  $SM$  and  $RM$  matrices.

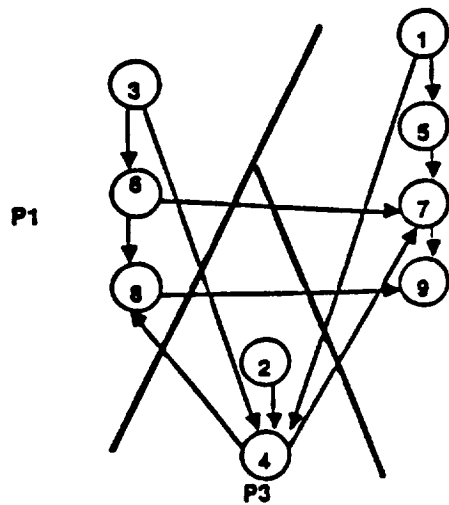


Figure 2. Partitioning a task graph into three blocks.

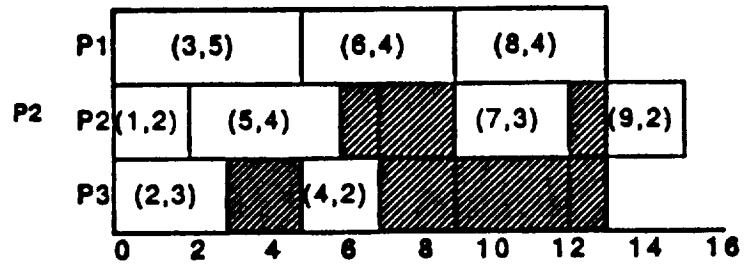


Figure 4. The Gantt chart for the Fig. 1 example.

#### Modules Description

$$T_1 = {}^0R_i$$

$$T_2 = p_i^* = {}^0R_i^T p_i^*$$

$$T_3 = z_{i-1} \dot{q}_i (1 - \lambda_i)$$

$$T_4 = \omega_i$$

$$T_5 = s_i = {}^0R_i^T s_i$$

$$T_6 = (z_{i-1} \ddot{q}_i + \omega_{i-1} \times z_{i-1} \dot{q}_i) (1 - \lambda_i)$$

$$T_7 = \dot{\omega}_i$$

$$T_8 = \dot{\omega}_i \times p_i^* + \omega_i \times (\omega_i \times p_i^*) + (z_{i-1} \ddot{q}_i + 2 \omega_{i-1} \times (z_{i-1} \dot{q}_i)) \lambda_i$$

$$T_9 = \ddot{p}_i$$

$$T_{10} = \ddot{r}_i$$

$$T_{11} = F_i$$

$$T_{12} = {}^i\omega_i = {}^iR_0 \omega_i$$

$$T_{13} = {}^i\dot{\omega}_i = {}^iR_0 \dot{\omega}_i$$

$$T_{14} = {}^iN_i = {}^iJ_i^T \dot{\omega}_i + {}^i\omega_i \times ({}^iJ_i^T \omega_i)$$

$$T_{15} = N_i$$

$$T_{16} = f_i$$

$$T_{17} = N_i + (p_i^* + s_i) \times F_i + p_i^* \times f_{i+1}$$

$$T_{18} = n_i$$

$$T_{19} = \tau_i$$

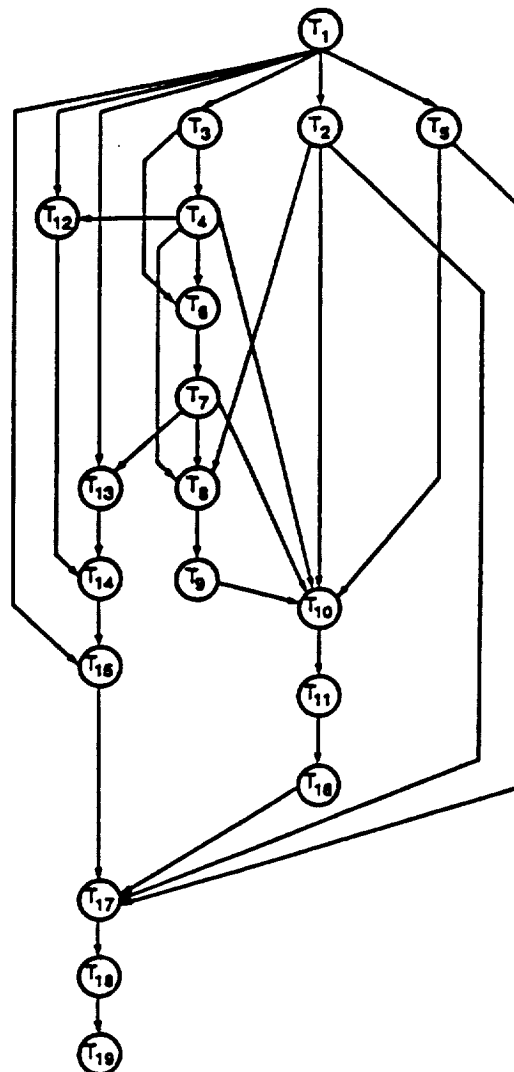


Figure 3. Task graph of Newton-Euler equations of motion.

**Table 1.** The PFT of the mapping-schedule in Fig. 1 example.

Insertion Stage	$PFT_1(n)$	$PFT_2(n)$	$PFT_3(n)$
1	2	2	2
2		6	6
3	7	7	7
4	10	10	
5		11	11
6		13	13
7		17	
8		20	
9		22	

**Table 2.** Cost value of each insertion stage for the Fig. 1 example.

$n$	$R(n)$	$P_{m0}(n)$	$p_{av}$	$R_1^{p*}(n)$	PFT	PCT
1	{1,2,3}	1,2,3	3	{3,2,1}	(5,2,3)	(0,0,0)
2	{ 5 }	2	1	{ 5 }	(5,6,3)	(0,0,0)
3	$\emptyset$	3	1	$\emptyset$	(5,6,5)	(0,0,0)
4	{4,6}	1,3	2	{6,4}	(9,6,7)	(0,0,0)
5	$\emptyset$	2	1	$\emptyset$	(9,7,7)	(1,1,5)
6	$\emptyset$	2,3	2	$\emptyset$	(9,9,9)	(1,1,5)
7	{7,8}	1,2,3	3	{8,7,idle}	(13,12,12)	(1,1,5)
8	$\emptyset$	2,3	2	$\emptyset$	(13,13,13)	(5,5,7)
9	{ 9 }	1,2,3	3	{9,idle,idle}	(13,15,7)	(5,5,7)
10						(6,7,7)

**Table 3.** Simulation results of the HM algorithm with different  $P/C$  ratios.

P/C	Optimality Percentage (%)		Relative Error (%)	
	Number of Processors = 2	Number of Processors = 3	Number of Processors = 2	Number of Processors = 3
10	71.43	90.48	3.05	0.44
5	71.43	90.48	3.67	0.29
2	71.43	95.24	5.61	0.05
1	52.38	66.67	13.47	3.09
0.5	30.77	15.38	17.34	16.65
0.2	0	0	102.02	92.26
0.1	0	0	224.32	254.36

**Table 5.** Comparison between simulation and implementation results for the eight-processor case.

Processor Number	Simulation Result	Hypercube Result	Relative Error (%)
0	379.70 ms	361.53 ms	4.79
1	330.08 ms	309.85 ms	6.13
2	383.12 ms	359.46 ms	6.17
3	428.04 ms	415.10 ms	3.02
4	406.98 ms	400.10 ms	1.70
5	376.50 ms	360.42 ms	4.27
6	376.16 ms	354.28 ms	5.82
7	374.50 ms	355.36 ms	5.11

**Table 4.** Comparison between simulation and implementation results for the four-processor case.

Processor Number	Simulation Result	Hypercube Result	Relative Error (%)
0	704.82 ms	679.71 ms	3.56
1	701.42 ms	676.69 ms	3.53
2	706.52 ms	675.78 ms	4.35
3	703.12 ms	669.23 ms	4.82



# PARALLEL ALGORITHMS FOR COMPUTATION OF THE MANIPULATOR INERTIA MATRIX

Masoud Amin-Javaheri and David E. Orin

Department of Electrical Engineering  
The Ohio State University  
Columbus, Ohio 43210

## Abstract

This paper presents the development of an  $O(\log_2 N)$  parallel algorithm for the manipulator inertia matrix. It is based on the most efficient serial algorithm which uses the composite rigid body method. Recursive doubling is used to reformulate the linear recurrence equations which are required to compute the diagonal elements of the matrix. It results in  $O(\log_2 N)$  levels of computation. Computation of the off-diagonal elements involves  $N$  linear recurrences of varying-size and a new method, which avoids redundant computation of position and orientation transforms for the manipulator, is developed. The  $O(\log_2 N)$  algorithm is presented in both equation and graphic forms which clearly show the parallelism inherent in the algorithm.

## 1. Introduction

A major problem in effectively realizing advanced control schemes for robotic systems has been the difficulty of implementing the kinematic and dynamic equations required for coordination, in real time. This problem is accentuated by the increasing structural and task complexities of the next generation of robots under development for space applications. Coordination of multiple-chain systems, with compliant structures operating in higher speeds regimes while making and breaking contact with the environment, places stringent computational demands on the control system.

One approach that has been used in advanced dynamic control schemes to obtain better performance has been to employ the inertia matrix to decouple the dynamics along the several axes of a robot manipulator. This allows either linear or nonlinear control schemes to be more effectively applied [1,2,3]. Specific tasks in which the inertia matrix has been applied in the control include surface tracking and object identification using force control [4] and computation of collision effects [5].

Determination of the inertia matrix involves a considerable amount of computation (approximately equal to that of Inverse Dynamics for a 6 degree-of-freedom manipulator). The most efficient serial algorithm for computing the inertia matrix was first developed by Walker and Orin [6] and requires  $O(N^2)$  time on a single processor system. Systolic architectures have been proposed in [7] which reduce the order of the computation to  $O(N)$  using  $N$  processors. The composite rigid body method developed in [6] was used in [7]. Essentially, sets of links at the end of the manipulator are considered to be fixed with respect to each other so that elementary physics principles may be used to compute their composite mass, center of mass, and moment of inertia. Use of the Newton-Euler dynamic equations on the reduced system results in efficient computation of the inertia matrix components.

This paper presents the development of a parallel algorithm to compute the manipulator inertia matrix in  $O(\log_2 N)$  time. Recursive doubling [8], which may be applied to linear recurrence equations to reduce the order of the computation, is used to compute the diagonal elements of the inertia matrix. Computation of the off-diagonal elements involves solution of  $N$  sets of linear recurrence equations of size  $N$ ,  $N - 1$ , etc. for which recursive doubling is not easily applied. Calculation of position and orientation transforms across the links of a varying-size composite rigid body at the base end of the manipulator is required. A new method is developed to compute the off-diagonal elements in  $O(\log_2 N)$  time, and it avoids redundant computation of the position and orientation transforms.

Set notation is used to develop the equations for the algorithm in a form which explicitly shows the parallelism available. The notation used was first developed in part in [9] where recursive doubling was used

to compute the Jacobian.

Previously, recursive doubling was applied to solve Inverse Dynamics in  $O(\log_2 N)$  time [10]. Prior to this, Lathrop [11] had achieved similar results through a logarithmic recursion method which was derived through a restructuring of the fundamental computational framework for the equations. Parallel computation of the inertia matrix has also been considered in the context of computing robot forward dynamics [12]. Recursive doubling was used to compute the diagonal elements and a modified row-sweeping algorithm was used to compute the off-diagonal elements with a resulting algorithm which was of  $O(N)$ . The work of this paper differs from [12] in that here quantities are not transformed to base coordinates before applying recursive doubling. Also, the new method for computing the off-diagonal elements results in a parallel algorithm which is of  $O(\log_2 N)$ . More recently, Fijany and Bejczy [13] have developed two parallel algorithms which achieve the lower bound in the computation time of  $O(\log_2 N) + O(1)$ . However, when they mapped the algorithms onto arrays of processors, they concluded that an  $O(N)$  parallel/pipeline algorithm will have a much improved efficiency with only a slight reduction in speedup.

In the next section, a brief overview of the  $O(N)$  parallel algorithm is given. In the section following, the  $O(\log_2 N)$  parallel algorithm is developed. The entire parallel algorithm is then summarized in a table in a form which shows much of the parallelism inherent in the algorithm. Finally, the work is summarized, conclusions are made, and several areas in which the work may be extended are discussed.

## 2. $O(N)$ Parallel Algorithm for the Inertia Matrix

An  $O(N)$  parallel algorithm, based upon the determination of the mass, center of mass, and moment of inertia of a series of composite rigid bodies for an  $N$ -degree-of-freedom open-chain manipulator, has been previously derived to compute the inertia matrix,  $\mathbf{H}(\mathbf{q})$  [7]. It was based on the earlier work of Walker and Orin [6] in which an efficient  $O(N^2)$  algorithm was developed for the inertia matrix to further realize efficient dynamic simulation on a single processor. In addition to the  $O(N)$  algorithm, various systolic architectures were also proposed in [7] to achieve a real-time response. A complete listing of the algorithm is shown in Table 1.

Briefly, Inverse Dynamics is applied to the manipulator  $N$  times. Starting with joint  $N$  and working toward joint 1, a unit acceleration is applied to a joint with all joint velocities and other joint accelerations equal to zero. This simply divides the manipulator into two sets of composite rigid bodies with one degree of freedom between them. The mass ( $M_i$ ), center of mass ( $\mathbf{c}_i$ ), and moment of inertia ( $\mathbf{E}_i$ ) for the composite rigid body at the end of the manipulator (links  $i$  through  $N$ ) are first computed recursively using basic physics principles. Then for each composite rigid body, the forces and moments at joint  $i$  due to a unit acceleration there ( $\mathbf{f}_{i,i}$ ,  $\mathbf{n}_{i,i}$ ) may be simply computed by applying the Newton-Euler equations of motion to the composite body. The component of the force along (prismatic) or moment about (revolute) the joint axis ( $i$ ) is the diagonal component of the inertia matrix,  $H_{i,i}$ . The required force or moment needed to ensure zero velocity and acceleration at joint  $j$  (for  $j < i$ ) is simply the off-diagonal element of the inertia matrix,  $H_{j,i}$ . These forces and moments ( $\mathbf{f}_{j,i}$ ,  $\mathbf{n}_{j,i}$ ) are recursively computed for the various joints of the lower composite rigid body by simple resolution of the force and moment at joint  $i$  to the required points. Only the diagonal and upper off-diagonal elements of the inertia matrix are computed since the inertia matrix is symmetric.

Noting Table 1, the computation of the composite rigid body parameters and the diagonal elements of the inertia matrix is seen to require  $O(N)$  time. The computation of the off-diagonal elements involves  $N$  recursions each of which may be computed in parallel, also giving  $O(N)$  time.

## 3. Development of an $O(\log_2 N)$ Parallel Algorithm for the Inertia Matrix

The concept of recursive doubling [8] may be used to develop an  $O(\log_2 N)$  parallel algorithm to compute the composite rigid body parameters and diagonal elements of the inertia matrix. It has been previously applied to achieve  $O(\log_2 N)$  parallel algorithms for Inverse Dynamics [10,11]. In order to understand the basic concept and develop the notation used, computation of the composite rigid body masses for a manipulator of eight degrees of freedom will first be considered in this section. A parallel algorithm of  $O(\log_2 N)$  will be developed for computing these masses,  $M_i$ . Then, the approach will be extended to computing all of the composite rigid body parameters, resulting in an  $O(\log_2 N)$  parallel algorithm for computing the diagonal elements of the

Table 1:  $O(N)$  Parallel Algorithm for Computing the Inertia Matrix.

CONST

$$\begin{aligned} M_{N+1} &= 0 \\ c_{N+1} &= 0 \\ E_{N+1} &= 0 \\ z_0 &= [0 \ 0 \ 1]^T \\ \sigma_i &= \begin{cases} 1 & \text{revolute joint} \\ 0 & \text{prismatic joint} \end{cases} \end{aligned}$$

BEGIN

{\* Computation of composite rigid body parameters and diagonal elements of the inertia matrix \*}

FOR  $i := N$  TO 1 DO

BEGIN1

$$\begin{aligned} M_i &:= M_{i+1} + m_i \\ c_i &:= \left\{ \frac{1}{M_i} [m_i (s_i^*) + M_{i+1} ({}^i U_{i+1} c_{i+1} + {}^i p_{i+1}^*)] \right\} \\ E_i &:= {}^i U_{i+1} E_{i+1} {}^{i+1} U_i + M_{i+1} [ ({}^i U_{i+1} c_{i+1} + {}^i p_{i+1}^* - c_i) \cdot ({}^i U_{i+1} c_{i+1} + {}^i p_{i+1}^* - c_i) 1 \\ &\quad - ({}^i U_{i+1} c_{i+1} + {}^i p_{i+1}^* - c_i) ({}^i U_{i+1} c_{i+1} + {}^i p_{i+1}^* - c_i)^T ] \\ &\quad + I_i + m_i [(s_i^* - c_i) \cdot (s_i^* - c_i) 1 - (s_i^* - c_i) (s_i^* - c_i)^T] \\ F_i &:= \sigma_i (z_0 \times M_i c_i) + \sigma_i (M_i z_0) \\ N_i &:= \sigma_i (E_i z_0) \\ f_{i,i} &:= F_i \\ n_{i,i} &:= N_i + c_i \times F_i \\ H_{i,i} &:= \sigma_i (n_{i,i} \cdot z_0) + \sigma_i (f_{i,i} \cdot z_0) \end{aligned}$$

END1

{\* Computation of off-diagonal elements of the inertia matrix \*}

FOR ALL  $i := N$  TO 1 DO

FOR  $j := i-1$  TO 1 DO

BEGIN2

$$\begin{aligned} f_{j,i} &:= {}^j U_{j+1} f_{j+1,i} \\ n_{j,i} &:= {}^j U_{j+1} (n_{j+1,i} + {}^j p_{j+1}^* \times f_{j+1,i}) \\ H_{j,i} &:= \sigma_j (n_{j,i} \cdot z_0) + \sigma_j (f_{j,i} \cdot z_0) \end{aligned}$$

END2

END

inertia matrix.

Computation of the off-diagonal elements of the inertia matrix involves  $N$  independent recursions of varying size for which recursive doubling is not easily applied. The last part of this section gives an  $O(\log_2 N)$  algorithm to compute these elements. It effectively uses the position and orientation transforms for the full  $N$ -link manipulator, which are computed while obtaining the diagonal elements, to transform forces and moments over a reduced, varying-size composite rigid body at the base end.

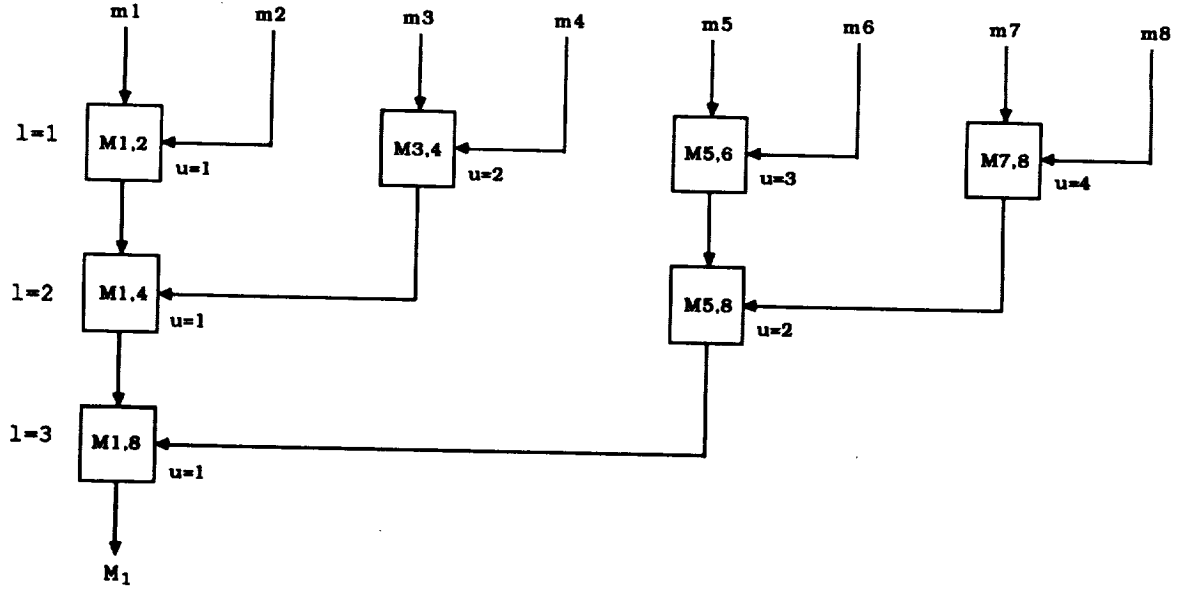


Figure 1: Flow of Data and Computation for Determining  $M_1$  ( $M_{1,8}$ )

### 3.1 Parallel Algorithm for $M_i$

As shown in Table 1, a recursive algorithm for computing  $M_i$  is given by

$$M_{N+1} = 0 \quad (1)$$

$$M_i = M_{i+1} + m_i \quad \text{for } i = N, \dots, 1. \quad (2)$$

Eq. (2) is an example of a linear recurrence relationship for which recursive doubling [8] may be applied. Consider computation of  $M_1$ , the composite rigid body mass for the entire manipulator. The mass across sets of two links may first be computed, all in parallel. For eight links, the links are simply combined as follows:

$$\{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\}, \{7\}, \{8\} \longrightarrow \{1, 2\}, \{3, 4\}, \{5, 6\}, \{7, 8\} \quad (3)$$

which indicates that the size of the set, for which the mass has been computed, has doubled. Again, doubling the number of links in a set gives

$$\{1, 2\}, \{3, 4\}, \{5, 6\}, \{7, 8\} \longrightarrow \{1, 2, 3, 4\}, \{5, 6, 7, 8\}, \quad (4)$$

indicating that the mass is now computed for sets of four links. The doubling effect may be recursively applied (recursive doubling):

$$\{1, 2, 3, 4\}, \{5, 6, 7, 8\} \longrightarrow \{1, 2, 3, 4, 5, 6, 7, 8\}, \quad (5)$$

so that the total mass of all eight links,  $M_1$ , is available after 3 steps ( $\log_2 8$ ).

Fig. 1 shows the flow of the data and computation involved in the three steps for computing  $M_1$  ( $M_{1,8}$ ). In the figure,  $M_{j,k}$  represents the mass of links  $j$  through  $k$ . This implies the following mapping relationship:

$$M_i \longrightarrow M_{i,N} \quad (6)$$

and

$$m_i \longrightarrow M_{i,i} \quad (7)$$

to extend the previous notation used.

Equations may be developed for computing  $M_1$ . The total number of levels of computation is given by

$$l_T = \log_2 N. \quad (8)$$

The total number of sets of links at each level whose composite mass is to be computed in parallel is given by

$$u_T = 2^{l_T - l} \quad (9)$$

where  $l$  is the level number. Also, the variable  $u$  is defined and used in Fig. 1 as the number of a set on a given level.

At the first level ( $l = 1$ ), links are combined in groups of two; at the second level ( $l = 2$ ), links are combined in groups of four, etc. Thus, the number of links in a set at each level, the width  $w$ , is a function of  $l$  and is given by

$$w = 2^l. \quad (10)$$

Each computational step in Fig. 1 then combines two sets of links into one:

$$\{i + 1, \dots, j\}, \{j + 1, \dots, k\} \longrightarrow \{i + 1, \dots, j, j + 1, \dots, k\} \quad (11)$$

where

$$i = w(u - 1), \quad (12)$$

$$j = w(u - 0.5), \text{ and} \quad (13)$$

$$k = w u. \quad (14)$$

Using the above notation for indexing, the following set of equations formalizes the parallel computation of  $M_1$  ( $M_{1,s}$ ).

$$\left\{ \begin{array}{l} l_T := \log_2 N \\ \text{FOR } l := 1 \text{ TO } l_T \text{ DO} \\ \quad \left\{ \begin{array}{l} w := 2^l \\ u_T := 2^{l_T - l} \end{array} \right\} \\ \quad \text{FOR ALL } u := 1 \text{ TO } u_T \text{ DO} \\ \quad \quad \left\{ \begin{array}{l} i := w(u - 1) \\ j := w(u - 0.5) \\ k := w u \end{array} \right\} \\ \quad \quad \text{BEGIN} \\ \quad \quad \quad M_{i+1,k} := M_{i+1,j} + M_{j+1,k} \\ \quad \quad \text{END} \end{array} \right\} \quad (15)$$

The above equations only determine the following set of composite rigid body masses:

$$\{M_{1,s}, M_{5,s}, M_{7,s}, M_{8,s}\} \equiv \{M_1, M_5, M_7, M_8\}. \quad (16)$$

To obtain the other composite rigid body masses needed, in general, multiple computations must be performed on each set of links at each level. All necessary computational steps are shown in Fig. 2. Note that the masses for subsets of links, from a link in the first half of a set to the last link in the set (Eq. (11)), must be computed. In so doing, another parameter which gives the total number of computations for each set of links on level  $l$  may be defined:

$$v_T = 2^{l-1}. \quad (17)$$

Here,  $v$  is also defined to be the computation number for set  $u$  for a given level  $l$ . Note that

$$u_T * v_T = 2^{l_T - l} * 2^{l-1} = 2^{l_T - 1} = \frac{N}{2} \quad (18)$$

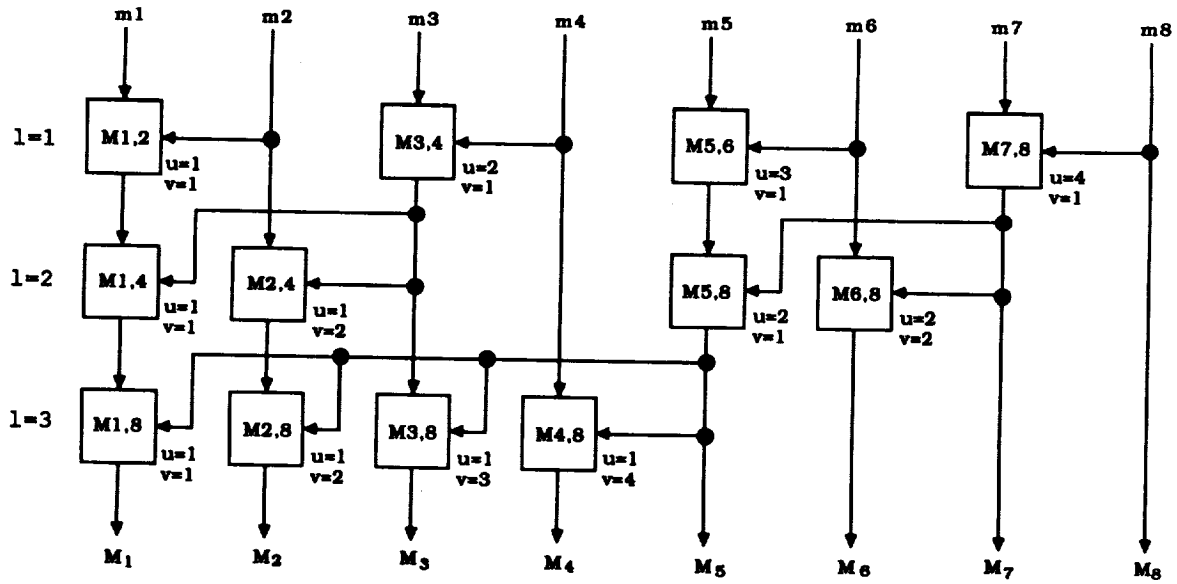


Figure 2: Flow of Data and Computation for Determining  $M_i$  ( $M_{i,N}$ )

which is constant over all the levels.

### 3.2 Parallel Algorithm for the Diagonal Elements

The approach taken to compute the composite rigid body masses,  $M_i$ , may be extended to develop the  $O(\log_2 N)$  parallel algorithm for all of the composite rigid body parameters and to further compute the diagonal elements of the inertia matrix. The algorithm is summarized in the first part of Table 2 and will be discussed in the following paragraphs.

First consider the computation of the composite rigid body parameters: mass ( $M$ ), center of mass ( $c$ ), and moment of inertia ( $E$ ). At a given level  $l$ , the main body of the computation is to calculate these in parallel, for any combination of  $u$  and  $v$ , across the sets of links from  $i+v$  to  $k$ . Note that the components of  $c$  and  $E$  are determined with respect to the coordinate system of the base link of the set ( $i+v$ ) so that the orientation and position transforms from links  $i+v$  to  $j+1$ ,  ${}^{i+v}U_{j+1}$  and  $\mathbf{p}_{i+v,j+1}$ , are required in the computations. Note also that transform of inertial quantities associated with the links, back to the base of the manipulator, is not required here as has been the case in other work [12]. As in the general case for recursive doubling [8], the transforms needed on level  $l$  are computed on level  $l-1$ . Also, in the equations, note that  $\mathbf{p}_{i,i+1} \equiv {}^i\mathbf{p}_{i+1}$ ,  $M_{i,i} \equiv m_i$ ,  $c_{i,i} \equiv s_i^*$ , and  $E_{i,i} \equiv I_i$  which are all initially given for each link  $i$ .

The ceiling function  $\lceil \cdot \rceil$  and conditions on the range of the indices  $j$  and  $k$  have been used so that the equations are appropriate for any number of degrees of freedom,  $N$ . Note that the computation is required only if the number of degrees of freedom  $N$  is greater than or equal to the number of the first link ( $j+1$ ) of the second of the sets of links to be combined (Eq. (11)).

The composite rigid body parameters as well as the diagonal components of the inertia matrix may be computed in  $O(\log_2 N)$  time and this is graphically depicted as Stage A of the computation in Fig. 3. In the figure, the numbers in the parentheses within a box give the associated links for which the computation is made. A "zeroth" level of computation is also shown in which the position and orientation transforms across each individual link are computed. Note that the angle for the first degree of freedom is not needed and that  ${}^iU_{i+1}$  and  ${}^i\mathbf{p}_{i+1}$  are computed for link  $i$ .

Table 2:  $O(\log_2 N)$  Parallel Algorithm for Computing the Inertia Matrix.

BEGIN11

$l_T := \lceil \log_2 N \rceil$

FOR  $l := 1$  TO  $l_T$  DO  
BEGIN12

$$\left\{ \begin{array}{l} w := 2^l \\ u_T := 2^{l_T-l} \\ v_T := 2^{l-1} \end{array} \right\}$$

FOR ALL  $u := 1$  TO  $u_T$  AND  
FOR ALL  $v := 1$  TO  $v_T$  WITH

$$\left\{ \begin{array}{l} i := w(u-1) \\ j := w(u-0.5) \\ k := wu \\ \text{IF } (k > N) \text{ THEN } (k := N) \end{array} \right\}$$

IF  $(j+1 \leq N)$  THEN DO  
BEGIN13

$${}^{i+v}U_{k+1} := {}^{i+v}U_{j+1} {}^{j+1}U_{k+1}$$

$$\mathbf{P}_{i+v,k+1} := \mathbf{P}_{i+v,j+1} + {}^{i+v}U_{j+1} \mathbf{P}_{j+1,k+1}$$

$$\mathbf{M}_{i+v,k} := \mathbf{M}_{i+v,j} + \mathbf{M}_{j+1,k}$$

$$\mathbf{c}_{i+v,k} := \frac{1}{\mathbf{M}_{i+v,k}} [\mathbf{M}_{i+v,j} \mathbf{c}_{i+v,j} + \mathbf{M}_{j+1,k} ({}^{i+v}U_{j+1} \mathbf{c}_{j+1,k} + \mathbf{P}_{i+v,j+1})]$$

$$\begin{aligned} \mathbf{E}_{i+v,k} := & {}^{i+v}U_{j+1} \mathbf{E}_{j+1,k} {}^{j+1}U_{i+v} + \mathbf{E}_{i+v,j} \\ & + \mathbf{M}_{j+1,k} [({}^{i+v}U_{j+1} \mathbf{c}_{j+1,k} + \mathbf{P}_{i+v,j+1} - \mathbf{c}_{i+v,k}) \cdot ({}^{i+v}U_{j+1} \mathbf{c}_{j+1,k} + \mathbf{P}_{i+v,j+1} - \mathbf{c}_{i+v,k})^T] \\ & - ({}^{i+v}U_{j+1} \mathbf{c}_{j+1,k} + \mathbf{P}_{i+v,j+1} - \mathbf{c}_{i+v,k}) ({}^{i+v}U_{j+1} \mathbf{c}_{j+1,k} + \mathbf{P}_{i+v,j} - \mathbf{c}_{i+v,j})^T \\ & + \mathbf{M}_{i+v,j} [(\mathbf{c}_{i+v,j} - \mathbf{c}_{i+v,k}) \cdot (\mathbf{c}_{i+v,j} - \mathbf{c}_{i+v,k})^T] \\ & - (\mathbf{c}_{i+v,j} - \mathbf{c}_{i+v,k}) (\mathbf{c}_{i+v,j} - \mathbf{c}_{i+v,k})^T \end{aligned}$$

END13

END12  
FOR ALL  $i := 1$  TO  $N$  DO  
BEGIN14

$$\mathbf{f}_{i,i} := \sigma_i(\mathbf{z}_0 \times \mathbf{M}_{i,N} \mathbf{c}_{i,N}) + \sigma_i(\mathbf{M}_{i,N} \mathbf{z}_0)$$

$$\mathbf{n}_{i,i} := \sigma_i(\mathbf{E}_{i,N} \mathbf{z}_0) + \mathbf{c}_{i,N} \times \mathbf{f}_{i,i}$$

$$\mathbf{H}_{i,i} := \sigma_i(\mathbf{n}_{i,i} \cdot \mathbf{z}_0) + \sigma_i(\mathbf{f}_{i,i} \cdot \mathbf{z}_0)$$

END14

END11

### 3.3 Parallel Algorithm for the Off-Diagonal Elements

To obtain the off-diagonal elements of the inertia matrix, the force and moment at joint  $i$  due to a unit acceleration there ( $\mathbf{f}_{i,i}$  and  $\mathbf{n}_{i,i}$ ) should be resolved to the previous joints ( $\mathbf{f}_{j,i}$  and  $\mathbf{n}_{j,i}$ ) for  $j \leq i$ . This involves a total of  $N$  linear recurrences of size  $N$ ,  $N-1$ , etc. Several approaches to parallel computation of the off-diagonal elements are first discussed in this section. Then an efficient approach, which uses the transforms computed in Stage A for the diagonal elements and which achieves an  $O(\log_2 N)$  time, is detailed.

First of all, computation of the off-diagonal terms may be computed in  $O(N)$  time if the parallel algorithm given in Table 1 is employed. This results in a computation time of  $K_1 O(N) + K_2 O(\log_2 N)$  for the entire inertia matrix where the  $K_1$  coefficient is relatively small. This is similar to the modified row-sweeping algorithm

Table 2 (continued)

```

BEGIN21
  FOR ALL  $x := 1$  TO  $N$  DO
    BEGIN22
       $l_x := \lceil \log_2 x \rceil$ 
      FOR  $l := 1$  TO  $l_x$  DO
        BEGIN23
          
$$\left\{ \begin{array}{l} w := 2^l \\ u_T := 2^{l_x-l} \\ v_T := 2^{l-1} \end{array} \right\}$$

          FOR ALL  $u := 1$  TO  $u_T$  AND
          FOR ALL  $v := 1$  TO  $v_T$  WITH
            
$$\left\{ \begin{array}{l} i := w(u-1) \\ j := w(u-0.5) \\ k := wu \end{array} \right\}$$

            IF  $(j+1 \leq x \leq k)$  THEN DO
              BEGIN24
                 $f_{i+v,x} := {}^{i+v}U_{j+1} f_{j+1,x}$ 
                 $n_{i+v,x} := {}^{i+v}U_{j+1} (n_{j+1,x} + p_{i+v,j+1} \times f_{j+1,x})$ 
                 $H_{i+v,x} := \sigma_{i+v} (n_{i+v,x} \cdot z_0) + \sigma_{i+v} (f_{i+v,x} \cdot z_0)$ 
              END24
            END23
          END22
        END21

```

applied by Lee and Chang [12] when computing the inertia matrix for parallel forward dynamics computation.

The order of the computation may be further reduced if recursive doubling is applied to each of the recurrences. Computation of the largest recurrence, which gives the elements of the last column of the inertia matrix, can then be achieved in  $O(\log_2 N)$  time. Since each of the recurrences may be computed in parallel, the overall computation time for the off-diagonal elements is  $O(\log_2 N)$ . The major problem with this approach, however, is that the position and orientation transforms ( $U$ 's and  $p$ 's) across the links of a varying-size composite rigid body at the base end of the manipulator are computed independently. This results in redundant computation both within this stage of computation and with Stage A for the diagonal elements.

The most efficient method for computing the off-diagonal elements is to use the transforms computed in Stage A while yet completing the computation in  $O(\log_2 N)$  time. But it should be understood that these transforms were basically computed for a manipulator of size  $N$  only. However if the computational structure of Stage A is considered in Stage B for the off-diagonal elements, then the more efficient algorithm is achieved. Noting Fig. 3, computation of the off-diagonal elements of column 6 is shown. Judicious elimination of many of the computational blocks (shown with dashed boxes) results in effective use of the transforms available from Stage A while still achieving  $O(\log_2 N)$  time.

Note that the computation in the left part of Stage B is generally not needed. This is shown implemented in the equations, in the last part of Table 2, through appropriate conditions on the indices  $j$  and  $k$ . Essentially, the computation is not required unless the column number  $x$  falls within the range of the second of the sets of links to be combined (Eq. (11)). In Fig. 3, the numbers in parentheses within a box give the indices for the  $f$ 's and  $n$ 's which are calculated. Not explicitly shown in the figure is the flow of the position and orientation transforms from Stage A to Stage B which are required for the computation of the off-diagonal terms.

The  $O(\log_2 N)$  parallel algorithm is shown in its entirety in Table 2. When compared with Table 1, it may be noted that the total number of primitive operations has increased with the parallel algorithm. In general,



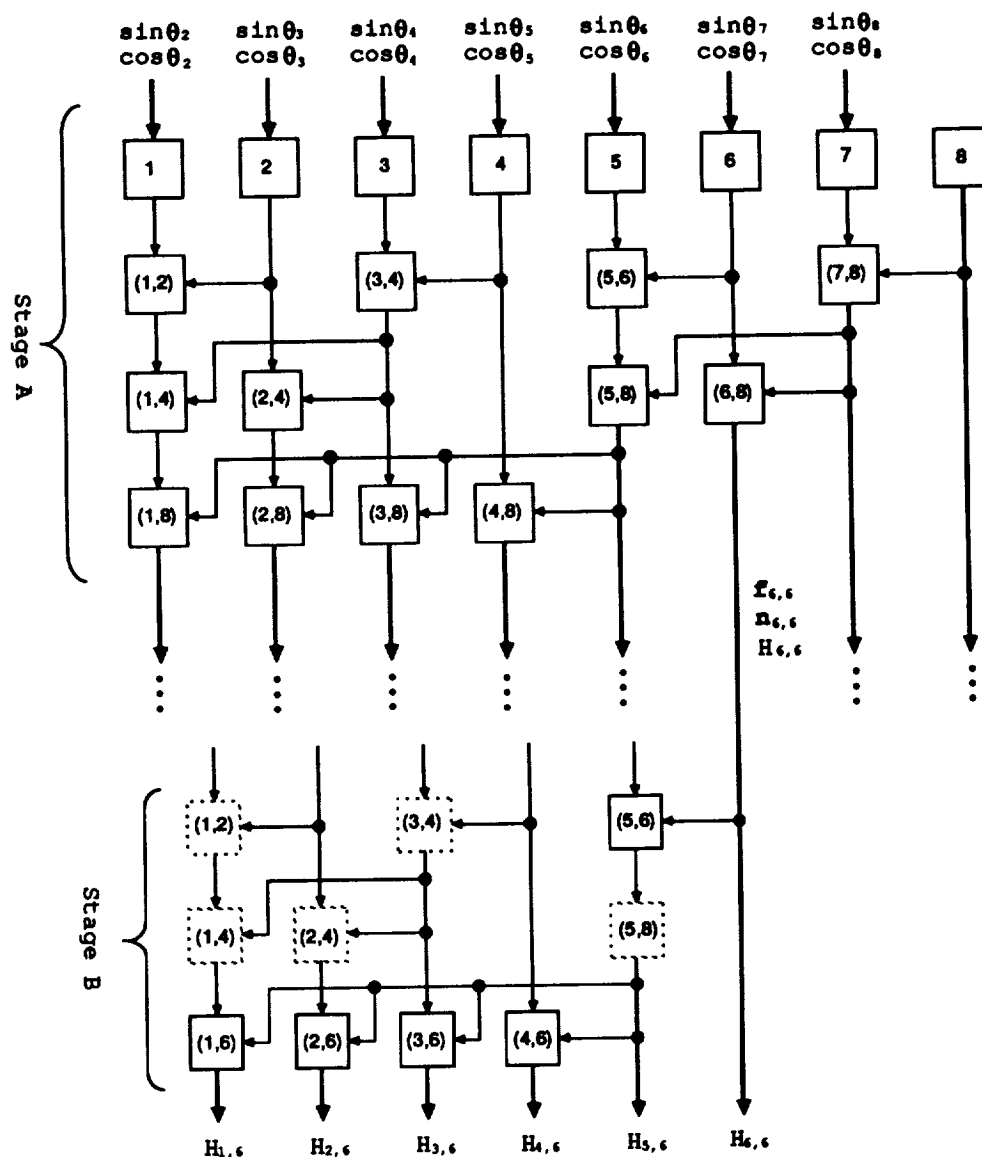


Figure 3: Flow of Data and Computation for the Parallel Algorithm ( $N = 8$ ).

the number of primitive operations increases as one attempts to decrease the order of the computation. In fact if the total number of operations decreased when going from a serial algorithm to a parallel algorithm, then the parallel algorithm should also be used on a serial processor since it becomes the most efficient serial algorithm as well.

#### 4. Summary and Conclusions

This paper has outlined the development of an  $O(\log_2 N)$  parallel algorithm for computing the  $N \times N$  inertia matrix for a robot manipulator. A listing of the algorithm is given in Table 2, and its flow of computation and data is shown in Fig. 3. In each case, the parallelism inherent in the algorithm is explicitly shown.

A recursive doubling technique [8] was used to achieve computational reduction over the  $O(N)$  parallel algorithm listed in Table 1 in computing the diagonal elements of the inertia matrix. It avoids transformation of inertial quantities, associated with each link, to base coordinates. An  $O(\log_2 N)$  algorithm, which uses

the position and orientation transforms computed for the entire manipulator when determining the diagonal elements, was then formulated to calculate the upper off-diagonal elements of the inertia matrix. Additional computation to determine the transforms over a varying-size composite rigid body (fixed set of links) at the base end of the manipulator is also avoided.

Investigations have also been made in associated work to determine the relationship between the order of the computation and the number of processors required for implementation. As expected, as the order of computation decreases, the total number of processors required increases.

Work is also under way to efficiently map the  $O(\log_2 N)$  algorithm into a parallel architecture structure while accounting for communications (I/O) overhead. The basic objective is to minimize the computational latency while maximizing CPU utilization. Further, work is under way to increase concurrent task processing on multiple processors by developing a new computational model which includes effective use of prediction algorithms. Hopefully, the work of this paper will provide the foundation for parallel implementations of the inertia matrix which will facilitate effective realizations of dynamic control schemes for space telerobotic systems.

### Acknowledgments

This work was supported by the National Science Foundation under Computational Engineering Grant No. EET-8718434.

### REFERENCES

- [1] J. R. Hewit and J. Padovan, "Decoupled feedback control of a robot and manipulator arms," in *Proc. of the 3rd CISM-IFTOMM Symposium on the Theory and Practice of Robots and Manipulators*, pp. 251-266, New York: Elsevier, 1979.
- [2] M. Leborgne, R. Dumas, J. J. Borrelly, C. Samson, and B. Espiau, "Nonlinear control of robot manipulators, Part 2: simulation and implementation of a robust control method," *IRISA/INRIA Report*, Rennes, France, 1986.
- [3] O. Khatib, "A unified approach for motion and force control of robot manipulators: the operational space formulation," *IEEE Journal of Robotics and Automation*, vol. RA-3, no. 1, pp. 43-53, February 1987.
- [4] J. Bay and H. Hemami, "Hybrid control of a robotic manipulator over an unknown constraint surface," To be Published in *IEEE Journal of Robotics and Automation*, 1988.
- [5] Y. Zheng and H. Hemami, "Mathematical modeling of a robot collision with its environment," *Journal of Robotics Systems*, vol. 2, no. 3, pp. 289-307, 1985.
- [6] M. W. Walker and D. E. Orin, "Efficient dynamic computer simulation of robotic mechanisms," *Journal of Dynamic Systems, Measurement, and Control*, vol. 104, pp. 205-211, September 1982.
- [7] M. Amin-Javaheri and D. E. Orin, "A systolic architecture for computation of the manipulator inertia matrix," in *Proc. of IEEE International Conference on Robotics and Automation*, pp. 647-653, Raleigh, North Carolina, April 1987.
- [8] P. M. Kogge and H. S. Stone, "A parallel algorithm for the efficient solution of a general class of recurrence equations," *IEEE Transactions on Computer*, vol. C-22, pp. 786-793, August 1973.
- [9] D. E. Orin, K. W. Olson, and H. H. Chao, "Systolic architectures for computation of the Jacobian for robot manipulators," in *Computer Architectures for Robotics and Automation*, J. H. Graham, Ed., pp. 39-67, New York: Gordon and Breach Science Publishers, 1987.
- [10] C. S. G. Lee and P. R. Chang, "Efficient parallel algorithm for robot Inverse Dynamics computation," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-16, no. 4, pp. 532-542, July/August 1986.
- [11] R. H. Lathrop, "Parallelism in manipulator dynamics," *The International Journal of Robotics Research*, vol. 4, no. 2, pp. 80-102, Summer 1985.
- [12] C. S. G. Lee and P. R. Chang, "Efficient parallel algorithms for robot forward dynamics computation," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. SMC-18, no. 2, pp. 238-251, March/April 1988.
- [13] A. Fijany and A. K. Bejczy, "Parallel algorithms for computation of manipulator inertia matrix," Engineering Memorandum, No. 347-88-249, Jet Propulsion Laboratory, July 1988.

# **SPATIAL REPRESENTATIONS AND REASONING**



# Planning robot actions under position and shape uncertainty

Christian LAUGIER\*  
LIFIA/IMAG  
46 Avenue Felix Viallet  
38031 Grenoble Cedex, FRANCE.

## Abstract

Geometric uncertainty may cause various failures during the execution of a robot control program. Avoiding such failures makes it necessary to reason about the effects of uncertainty in order to implement robust strategies. In this paper, we first point out that a manipulation program has to be faced with two types of uncertainty: those that might be locally processed using appropriate sensor based motions, and those that require a more global processing leading to insert new sensing operations. Then, we briefly describe how we have solved the two related problems in the SHARP<sup>1</sup> system: How to automatically synthesize a fine motion strategy allowing the robot to progressively achieve a given assembly relation despite position uncertainty? How to represent uncertainty and to determine the points where a given manipulation program might fail?

## 1 Introduction

### 1.1 Statement of the problem

A robot and its working space constitute a complex mechanical system that cannot be completely modelled. This means that both the environment and the actions executed by the robot cannot be exactly predicted at programming time, and that the resulting uncertainty may cause the program to fail. Consequently, it is necessary to determine the points where uncertainty is too high according to the required precision, and then to reduce this uncertainty using appropriate *sensory based strategies*. The purpose of this paper is to discuss this problem, and to answer as far as possible the two following questions: How to automatically synthesize a fine motion strategy (i.e. a manipulation strategy combining sensing operations with small robot movements) allowing the robot to progressively achieve a given assembly relation despite position uncertainty? How to represent uncertainty and to determine the points where a given manipulation program might fail?

The first question is related to the fact that some position errors can be directly taken into account by the system when planning contact based motions. The basic hypothesis in this case, consists in assuming that the *local* environment of the task can be considered as a "geometric guide" for the robot. The second question is motivated by the fact that some other position errors have a more *global scope*, since they are directly related to the interaction that exists between the actions of the manipulation program. For example, a grasping operation generating a too large uncertainty on the position of the chosen grasping points, may lead to a failure during the next part mating operation. The basic strategy in this case consists in first propagating the uncertainty terms through the program in order to determine possible failure points, and then to amend the program by inserting appropriate sensory based operations.

---

\*Senior Researcher at INRIA

<sup>1</sup>SHARP is an automatic robot programming system currently under development at LIFIA

## 1.2 Planning fine motion strategies

Several types of techniques have been developed for dealing with uncertainty in robot programming. Some of these techniques are aimed at executing "compliant motions" involving both force and position parameters in the command [23] [15] [8]. The other techniques were developed for the purpose of constructing complete fine motion strategies. A first approach for solving this problem consists in generating a solution by instantiating some predefined "procedure skeletons" using error bounds computations [19] [12] [14], or by assembling a set of partial strategies using learning techniques and expert rules [4]. The major limitation of this approach comes from the fact that it relies on the following hypothesis: any assembly operation can be unambiguously associated to a more general assembly class that can be processed using a single type of strategy. Unfortunately, a slight modification of the local geometry of the involved workpieces may drastically change the strategy to apply. This means that it seems impossible to identify a reasonable set of assembly classes.

A more general approach consists in constructing the fine motion strategies by reasoning on the geometry of the task [13] [6] [9] [22]. This approach leads to consider the local environment of the workpieces to assemble as a "geometric guide" for the robot. Then, planning a fine motion strategy may be seen as the determination of an ordered sequence of well chosen contacts and of intermediate situations. The formal bases of this approach are given in [13], and a first attempt of implementing it in a polygonal world is described in [6]. But one suspects a too high algorithmic complexity for making the problem manageable in real situations. This is why we have developed a method allowing to reduce the size of the search graph by heuristically guiding the geometric reasoning.

This method is briefly described in section 2. The basic idea consists in deducing a fine motion strategy from an analysis of the different ways in which the assembled parts may be theoretically dismantled. This analysis is executed on an explicit representation of the contact space. It leads to successively construct a *state graph* representing the set of potential solutions, and to search this graph in order to find a "good reverse path" defining a feasible fine motion program.

## 1.3 Reasoning on position uncertainty

This problem has already given rise to several theoretical and practical developments aimed at achieving two different goals: (1) dealing with position uncertainty when programming a manipulation robot [19] [1] [14] [17], and (2) combining the uncertain data provided by sensors while updating or constructing the model of the robot environment [3] [18] [7] [5]. In spite of their different formulations, these problems have led to the development of similar approaches for reasoning on position uncertainty (although the applied computational models are different). The basic mathematical tools involved in this reasoning are described in [16] and [17].

Work done in the context of robot programming is aimed at explicitly representing the error bounds associated to the position variables of a manipulation program, in order to propagate them through the model and to compute their values in any point of the program. This approach was initially devised for computing the values of the parameters associated to a set of predefined manipulation procedures [19] [14]. We will see further how it can be applied for evaluating the correctness of a manipulation program, according to the precision constraints imposed by the task. Some other researchers have considered the problem of reasoning on position uncertainty as a computational subproblem of the interpretation of sensory data. In order to avoid as far as possible to over-estimate the errors when combining such data, the developed approaches have been based on statistical models [18] [5] [7].

The statistical approach is probably well adapted to sensor fusion. But its computational characteristics seem to limit its applicability domain to rather simple situations involving very few contacts. Unfortunately, assembly programs include a great number of multiple contacts and of complex assembly relations. This is why we have first chosen to implement an approach based on error bounds

computations (see section 3.2). This approach was consistent with the main characteristics of our problem: checking for the correctness of a sequence of actions, very few propagation operations have to be applied for each assembly step, several terms of the initial uncertainty are quickly reduced by contacts. But a more recent implementation based on statistical models [17], has shown that the results obtained using the two approaches are very similar (in this particular context). This means that the choice of one of these models is not of a prime importance for us, and that this choice has no real consequences on our verification/ correction method.

## 2 Planning fine motion strategies

### 2.1 Outline of our approach

As mentioned above, the applied method for planning fine motion strategies leads to consider the local environment of the workpieces to assemble as a “geometric guide” for the robot. Then, planning a fine motion strategy may be seen as the determination of an ordered sequence of well chosen contacts. Each selected contact leads to decrease the “distance” to the goal situation by reducing the position uncertainty. It gives rise to the execution of a guarded compliant motion. All the parameters of the related motion command (motion direction and amplitude, force and termination conditions) are finally computed using various accessibility and reliability criteria.

More practically, our method *deduces a solution from an analysis of the different ways the assembled parts may be theoretically dismantled*. This approach has been motivated by the fact that the existing contacts constrain the relative movements of the two parts, and reduce this way the number of hypotheses to formulate. An important characteristic of our method is to progressively guide the search choices, by successively analysing more and more detailed constraints drawn from the geometry of objects. As we will see further, a practical way for doing that consists in separating the computation of potential reachable positions and valid movements, from the determination of those which are really executable by the robot. This approach has been implemented in our system by a two phases algorithm leading to successively construct a *state graph* representing the set of potential solutions, and to search this graph in order to find a “good reverse path” defining a feasible fine motion program:

- *The analysis phase* constructs the state graph by reasoning on a fictitious dismantling of the assembly. For that purpose, the system determines at each step the different contact situations which can be reached from the current situation by applying a single motion. Only the local moving constraints associated to the contacts are examined at this step. The applied method leads to progressively decrease the number of contacts.
- *The search phase* determines a “reverse path” in the graph, i.e. a path starting from a node having an empty set of contact and ending at the node corresponding to the final assembly. This search phase is based on heuristics which attempt to optimize the selected solution in terms of both efficiency (number of operations) and reliability (robustness of the selected motions). In case of failure –for example one contact situation cannot be achieved because of the control errors– the graph is locally refined by introducing some new potential motions and contacts [9].

This approach allows us to reduce the size of the search graph. It also leads to apply costly geometric computations only when intricate situations have to be processed. The applied method makes use of a symbolic representation of contacts which includes three types of information [11]: the geometric entities involved in the contact, the topology of the contact and the associated geometric parameters.

## 2.2 Reasoning on motion constraints

Each contact reduces the number of d.o.f of the moving object. In order to determine the next motion to execute from a given contact situation, the system must reason on the moving directions which are constrained by the contacts. Consequently, it is necessary to explicitly represent the valid movements which can be *locally* associated to a contact situation. In order to simplify the computations, we will consider that these movements are either pure translations or pure rotations which are defined independently of their possible amplitudes. If  $A$  is a mobile object in contact with  $B$ , we will define a potential motion for  $A$  as "*a motion having an amplitude greater than the maximum control error, and generating no collision between the features in contact*". In practice, this definition has led us to develop an analytic support allowing to explicitly represent the forbidden motions, those which preserve the contacts and those which break them [11]. The applied method leads to represent a set of possible translating motions as a particular domain on a unitary sphere. The intersection of such domains is computed using simple functions of the type  $\vartheta = \arctan((N_x \cdot \cos \varphi + N_y \cdot \sin \varphi) / -N_z)$ , where  $(N_x, N_y, N_z)$  is the normal external vector to the related contact plane [20].

Similar representations are also used for dealing with curved supports. But in this case, the completeness property of the representation is preserved by using a first order approximation, leading to locally represent the potential compliant motions as a set of tangential motions. Rotating motions are modelled using a different method leading to group together all the rotation axes which generate an "homogeneous behavior" relatively to the contacts (for example: rotations which maintain the topological properties of the contact).

## 2.3 Dealing with the state graph

The sets of contacts and of their associated potential motions are combined in order to construct the *state graph associated to the fine motions*. This graph represents all the combinations of motions and robot states which have been selected as *potential elements of solution* for the problem to be solved. It is represented by a directed graph  $G(A/B)$ , where each node represents a robot state  $E_p$ , and each arc defines a motion allowing the robot to move from one state to an other one [11].

Our analytic representation of potential motions allows us to characterize the whole set of movements which are potentially feasible from a given state  $E_p$ . But this representation cannot be directly used by the motion planner, since each constructed domain  $D$  represents an infinite set of possible solutions (and consequently an infinite set of possible arcs for each node in the state graph). A classical technique dealing with this problem, consists in *discretizing the sets of potential solutions*. This technique leads to split each domain  $D$  into a finite set of small spherical domains of the type  $\Delta\varphi \times \Delta\vartheta$ . Each obtained domain  $\Delta S$  represents a set of motions which will be "globally" analysed by the system. This approach requires that all the motion directions in  $\Delta S$  allows to theoretically achieve the same symbolic contact situation, when executing these motions from a given position  $P$ . If the objects are polyedra and the motions are pure translations, such a constraint may be evaluated using a "visibility" analysis technique [2].

But the high algorithmic complexity of this approach along with its inability to deal with rotations and curved surfaces, has led us to make use of an heuristic based approach. The basic idea consists in analysing a subset of the possible solutions, by selecting in  $D$  the most promising motion directions. This approach is consistent with the fact that most of the required movements for mating two mechanical parts, are executed along some *privileged directions* defined by the contact surfaces. Then, the state graph may be constructed using the following algorithm:

1. Create the node  $GS$  and insert it in the list  $OPEN$ .
2. If  $OPEN = \emptyset$  then return  $G(A/B)$ , else process the node  $x$  located at the head of the list  $OPEN$ .



Choose  $n$  directions  $d_1, d_2 \dots d_n$  in  $D_x$ . Create an arc  $a_{xi}$  for each chosen direction  $d_i$ .

3. Create a node  $y_i$  for each arc  $a_{xi}$ . If the state  $E_i$  associated to  $y_i$  is already represented by a node  $z$  in  $G(A/B)$ , then merge  $y_i$  and  $z$ .
4. Insert the new nodes having an empty set of contacts in  $IS$ ; insert the other nodes in the list  $OPEN$ . Goto (2).

$OPEN$  represents the list of the next nodes to process, and  $D_x$  is the set of potential motions associated to the node  $x$ .  $GS$  is the goal state (the parts  $A$  and  $B$  are assembled), and  $IS$  is the set of the possible starting states for the fine motion strategies (states having an empty set of contacts). The geometric functions which have been developed for computing the parameters of the graph items (contacts, potential motions and sensory identification for each node; valid ranges of positions, sliding surfaces and sticking surfaces for each arc) are described in [10] and [11].

The moving directions are selected in  $D_x$  according to an heuristic function. For example, four directions will be initially generated by a couple of non-parallel planar contacts:  $d_1 = N_1 \wedge N_2$ ,  $d_2 = -d_1$ ,  $d_3 = d_1 \wedge N_1$  and  $d_4 = d_2 \wedge N_2$ , where  $N_1$  and  $N_2$  are the external normal vectors to the contact faces  $F_1$  and  $F_2$ .  $d_1$  and  $d_2$  define two compliant motions allowing to maintain the contacts;  $d_3$  and  $d_4$  define two motions leading to respectively break the contact associated to  $F_2$  and to  $F_1$ . These moving directions are considered by the system only if they are included in  $D_x$ .

## 2.4 Constructing a fine motion strategy

### 2.4.1 Searching the state graph

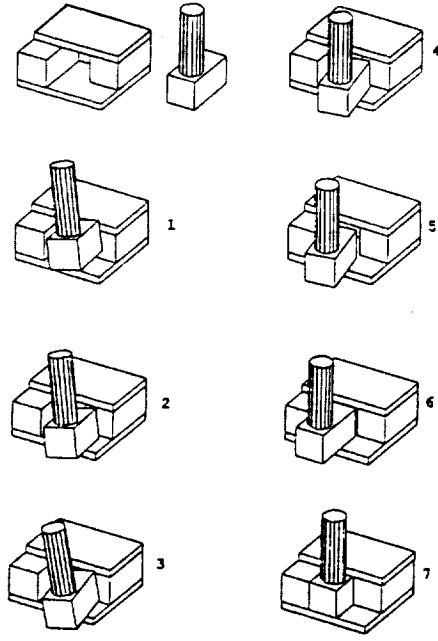
Let  $IS$  be the set of nodes of  $G(A/B)$  which have an empty set of contacts, and  $GS$  the state corresponding to the situation where  $A$  and  $B$  are assembled. Any path in  $G(A/B)$  starting from a node  $s$  in  $IS$  and ending at the node  $GS$ , may be considered as a fine motion strategy allowing to assemble  $A$  on  $B$ . Then searching for a solution in  $G(A/B)$  can be done using the following algorithm:

1. Search for a path  $SG$  starting from a node  $s$  in  $IS$  and ending at the node  $GS$ .
2. Verify that each arc in  $SG$  represents a feasible motion (no collision, reachability of the goal). Refine  $G(A/B)$  in case of failure, and goto (1).
3. Synthesize the fine program represented by  $SG$ .

Since the current version of the system discards the potential motions which may stop in different contact situations (because of control errors), each selected solution  $SG$  is represented by a single path in  $G(A/B)$  –and not by a “complete subgraph” as discussed in [10]–. Such paths are computed using a combination of a *cost function* and of a set of *dynamic advices* implemented using production rules [9] [20]. The cost function exploits the heuristic weights associated to the graph items, in order to both minimize the number of operations and to maximize the reliability of the selected motions. The dynamic advices are activated when some impractical situations are detected by the system (for example: adjacent contacts or closed obstacles). They lead to locally refine the graph. This approach allows us to reduce the algorithmic complexity by only exploring “in detail” the branches of the graph which are really significant according to the selected solution.

### 2.4.2 Synthesizing a fine motion program

Synthesizing a fine motion program from a path  $SG$  requires to first check for the validity (collision and reachability criteria) of each involved motion, and secondly to apply rewriting rules for generating the program. The validity tests are executed using the following computations [10]:



```

(RREALIZE-S
  (R-robot ON R0)
  (VIA C1 C2 ... Cn))

(RREALIZE-C
  (point-P1 ON face-B1)
  (ALONG (TRANSLAT :VECT vector-V1)))

(RREALIZE-C
  (edge-A1 ON face-B1)
  (ALONG (ROTATION :AXE (make-axe :PT point-P1 :VECT vector-V2)))
  (BY-MAINTAINING (point-P1 ON face-B1)))

(RREALIZE-C
  (face-F1 ON face-B1)
  (ALONG (ROTATION :AXE (make-axe :PT point-P1 :VECT edge-A1)))
  (BY-MAINTAINING (edge-A1 ON face-B1)))

(RREALIZE-C
  (edge-A2 ON face-B2)
  (ALONG (TRANSLAT :VECT vector-V3))
  (BY-MAINTAINING (face-F1 ON face-B1)))

(RREALIZE-C
  (face-F2 ON face-B2)
  (ALONG (ROTATION :AXE (make-axe :PT point-P2 :VECT edge-A2)))
  (BY-MAINTAINING (face-F1 ON face-B1)(edge-A2 ON face-B2)))

(RREALIZE-C
  (point-P3 ON face-B3)
  (ALONG (TRANSLAT :VECT edge-A1))
  (BY-MAINTAINING (face-F1 ON face-B1)(face-F2 ON face-B2)))

```

Figure 1: A fine motion strategy computed by the system.

$$\begin{aligned}
 \text{Sweep}(A, d) \cap \text{Gros}_\varepsilon(B) = \emptyset &\Rightarrow \text{no collision} \\
 A(p) \cap \text{Gros}_\varepsilon^{-1}(B) \neq \emptyset &\Rightarrow p \text{ is reachable}
 \end{aligned}$$

where  $\varepsilon = 2(\varepsilon_p + \varepsilon_i)$  and the terms  $\varepsilon_p$  and  $\varepsilon_i$  represent respectively the control and the sensing error bounds;  $\text{Sweep}(A, d)$  is the volume swept by  $A$  when moving along  $d$ ,  $\text{Gros}_\varepsilon(B)$  and  $\text{Gros}_\varepsilon^{-1}(B)$  respectively represent the obstacles  $B$  grown and shrunk according to  $\varepsilon$ , and  $A(p)$  is the object  $A$  in the configuration  $p$ . Then the missing motion parameters of each selected movement are computed, in order to synthesize a sequence of guarded compliant motions of the type:

MOVE  $\langle \text{objet-}A \rangle$  ALONG  $\langle T \rangle$  BY-MAINTAINING  $\langle C \rangle$  UNTIL  $\langle A \rangle$

where  $T$  and  $C$  are the symbolic motion parameters recorded in the graph, and  $A$  represents the set of contacts which may stop the movement. The numerical values associated to these parameters at the execution time are computed using the geometric model and some predefined thresholds. For example, a face belonging to  $A$  will generate a condition of the type " $F_v > \text{threshold}$ ", where  $F_v$  is the projection of the reaction force on the moving direction  $v$ , and  $v$  is assumed to be included in the friction cone (this condition is associated to the termination predicate). A similar computation is executed for the compliant parameters, but the needed thresholds are currently tuned by the operator.

### 3 Dealing with global uncertainty constraints

#### 3.1 Outline of our approach

The main problem to solve is to decide if a manipulation program produced by the system is guaranteed to work in the real world (according to the known world model), i.e. if the precision constraints imposed by the task are "compatible" with the error terms associated to the position variables of the program. This means that both nominal positions and error terms have to be computed in any point of the program, in order to be compared with their associated constraints. Since errors are modified

by robot actions, this computation must be executed using an appropriate *propagation mechanism*. Let us consider as an example an object which has been grasped and moved by the robot. Its initial position error have been reduced along some directions by the grasping operation, before beeing grown by a term representing the inaccuracy of the motion command. This type of computation is used by the system for both determining possible failure points and possible correction points. The related verification/ correction process operates in two modes [17]:

- In the verification mode, the system applies a *forward propagation mechanism* for computing the resulting uncertainties at any point of the program, in order to compare them with the uncertainty constraints imposed by the task.
- In case of failure (an uncertainty term does not verify the associated constraint), the system switches to the correction mode in order to determine the possible correction points and the type of corrective action to apply. For that purpose, it applies a *backward propagation mechanism* leading to compute the uncertainty constraints which should be verified in the precedent steps of the manipulation program, for avoiding the studied failure.

### 3.2 Modeling uncertainty

The *nominal position* of an object  $A$  is a theoretical value which is used for programming the robot, but which is never reached at execution time because of various positioning errors. The associated variable in the program is a geometric transform  $T_a$  verifying the relation  $R_a = Base \star T_a$ , where  $R_a$  is the frame associated to  $A$ , and  $Base$  is the reference frame of the robot workspace. Then, the gap existing between the nominal position of  $R_a$  and its real position, may be represented by a geometric transform  $\epsilon_a$  verifying the relation  $R_a^* = Base \star T_a \star \epsilon_a$ , where  $R_a^*$  represents the real position of  $R_a$ . The same property holds when considering the relative positions of two objects  $A$  and  $B$ .

As explained in section 1.3, we have chosen to represent position uncertainties using error bounds. Then, the uncertainty  $I_{ab}$  associated to the position of  $R_b$  relatively to  $R_a$ , is defined as the set of all possible errors  $\epsilon_{ab}$ . Each transform  $\epsilon_{ab}$  may be characterized by a triplet  $(t, u, \alpha)$ , where  $t$  is a translating vector in  $\mathbb{R}^3$ ,  $u$  is a unitary vector representing the rotation axis, and  $\alpha$  is the rotating angle. Then, the uncertainty  $I_{ab}$  may be seen as a subset  $E$  of the cartesian product  $\mathbb{R}^3 \times S(1) \times [-\pi + \pi]$ , where  $S(1)$  is the unitary sphere [17]. Since these subsets are generally difficult to compute, we will make use of approximations leading to first project  $E$  on each space  $\mathbb{R}^3$ ,  $S(1)$  and  $[-\pi + \pi]$ , and then to approximate the obtained sets  $Tr$ ,  $U$  and  $D$  using simple surrounding sets. This approach widely simplifies the involved computations. It is consistant with the verification/ correction scheme, which leads in this case to reason on the "worst case hypothesis". Using this approach, it becomes possible to represent an uncertainty by a set  $Tr \times U \times D$ , where  $Tr$  is either a sphere, a disc or a bounded straight line;  $U$  is either the unitary sphere  $S(1)$  or a vector;  $D$  is an interval  $[-\alpha + \alpha]$ . For example, the position uncertainty associated to an object lying on an horizontal plane  $z = a$ , is represented by a set of the type:  $Disc(z_o, \epsilon) \times \{vector z_o\} \times [-\alpha + \alpha]$ . In case of a vertical cylindrical contact, the first item (the disc) is replaced by an interval  $[b - \epsilon \ b + \epsilon]$  in the  $z$  direction ( $b$  is the nominal position of the object).

**Remark:** In the statistical approach, the uncertainty  $I_{ab}$  is modeled using the covariance matrix of  $\epsilon_{ab}$ , when the six parameters (translation and rotation vectors) of  $\epsilon_{ab}$  verify a gaussian law [17].

### 3.3 The world model

A *world state* is represented by a directed graph, where each node represents the reference frame associated to an object, and each arc denotes a geometric transform and its associated uncertainty term. The basic structure of this representation is a tree having the reference frame of the robot

workspace as root. This root usually represents the fixed base of the robot. Then, each arc defines either a spatial relation between two objects, or a physical relation created by a particular action of the robot (for instance: a contact between two objects or a functional link between a sensor and the sensed object). It is characterized by a couple  $\{T_{ab}, I_{ab}\}$ , where  $T_{ab}$  is a nominal geometric transform, and  $I_{ab}$  is the associated uncertainty.

This dynamic structure is fundamental for dealing with uncertainty, because it allows the computation of error terms at the points where they are really significant. For example, creating a contact between two objects  $A$  and  $B$  leads to reduce the *relative position uncertainty* of  $A$  and  $B$ . Then, it makes sense to explicitly represent this information in the world model, and to propagate its effects through the graph.

Lets consider the manipulation example shown in figure 2. The link  $Arc(R_o, R_a)$  have been suppressed in the world state  $W_3$ , because it represents a "wrong" way for computing  $I_{oa}$  which only depends in this case on the terms  $I_{ba}$  and  $I_{ob}$ . Conversely, a new link  $Arc(R_b, R_a)$  has been created after having computed  $T_{ba}$  and  $I_{ba}$ .  $T_{ba}$  represents the composition of several geometric transforms ( $T_{ba} = T_m^{-1} \star T_{ob}^{-1} \star T_{oa}$ , where  $T_m$  is the executed motion);  $I_{ba}$  is a similar expression, but it is computed using appropriate operators leading to combine sets of geometric transforms (these sets represent the uncertainty terms). In general, the related computations are difficult to implement [17]. Fortunately, the approximations which have been applied for representing uncertainty sets (spheres, discs and bounded straight lines), allow the implementation of simpler algorithms. For example, the composition of a sphere  $S(\epsilon_1)$  and of a disc  $D(axis, \epsilon_2)$  representing two translating uncertainties, will give rise to a new term represented by a sphere  $S(\epsilon_1 + \epsilon_2)$ .

Since world changes are caused by robot actions, two types of operators have to be developed for updating the world model: those devoted to the computation of the uncertainty associated to a combination of existing relations (*composition* of two uncertainties and *inversion* of an uncertainty), and those which are used for computing the uncertainty terms which have been modified by robot actions (*projection* and *fusion* operators). The projection operator is used for computing the terms which have been shrunk by a contact; the fusion operator is used for computing the resulting uncertainty when a sensing operation have been executed. For example, the resulting uncertainty  $I_{ba}$  in  $W_3$  is obtained by projecting the expression  $I_{ob}^{inv} \otimes I_{oa}$  on  $P \times N \times [-\pi + \pi]$ , where  $P$  is a plane parallel to the jaws of the gripper, and  $N$  is a direction normal to  $P$ ;  $\otimes$  and "inv" are respectively the composition and the inversion operators mentioned above.

### 3.4 Modeling robot actions

Propagating uncertainties through a manipulation program requires to precisely know the effects of robot actions (in terms of nominal positions and of associated uncertainties). Since the existing robot programming languages provide instructions which cannot be fully interpreted in these terms, we have developed a geometric based language aimed at avoiding such ambiguities [10]. All the same, the required "predictability" property is not verified for motion commands having several possible termination conditions, and for commands involving compliant motions. This is why we have chosen to only apply the verification/ correction scheme on linear sequences of operations of the following types: free space motions, motions aimed at achieving or at breaking a contact (this includes grasping and dropping operations), and sensing operations aimed at locating an object. Then, more complex constructions implementing fine motion strategies are supposed to be globally correct. For that purpose, they will be assimilated to "macro-actions" that can be used for achieving complex assembly relations (in this case, the final uncertainty is directly deduced from the known assembly clearances).

In our geometric language, the semantics associated to a robot action is defined by the modifications that this action applies to the world model: nominal positions and their associated uncertainties

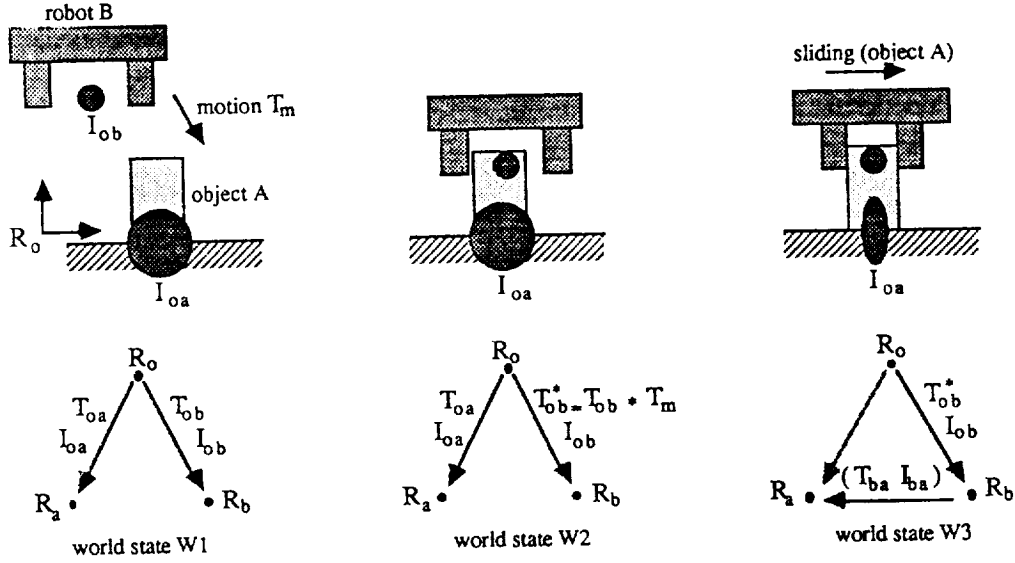


Figure 2: World model modifications generated by a grasping operation.

(nodes), spatial and physical relations (arcs). Free space motions lead to modify the position parameters, according to the executed movements and to the accuracy of the robot. Motions involving contacts lead to both modify the position parameters and to locally change the graph structure, as explained in section 3.3 (Figure 2 illustrates).

### 3.5 The verification/correction scheme

#### 3.5.1 Representing a constrained manipulation plan

According to the previous hypotheses, a manipulation plan may be seen as a linear sequence of actions  $A_1 A_2 \dots A_n$ . These actions lead to progressively move from an initial world state  $W_o$  to a final one  $W_n$ . Each world state  $W_i$  is represented by a graph as explained in section 3.3; its contents depends on both the last executed action  $A_i$  and the previous world state  $W_{i-1}$ . But the action  $A_i$  is guaranteed to produce the expected result (i.e. the world state  $W_i$ ), only if some conditions hold before its execution (i.e. in  $W_{i-1}$ ). For example, a grasping operation may fail if the uncertainty associated to the initial position of the object is too high relatively to the width of the jaws. Then, checking for the correctness of a manipulation plan necessitates to reason on such conditions. This means that the plan must contain an explicit representation of the *position constraints*  $CO_i$  that have to be verified in each world state  $W_i$ ,  $i = 0, n$ . Such constraints are associated to the position variables of the plan. They are expressed using couples of the type  $(T_{ab}^i = t_o, I_{ab}^i \prec i_o)$ , where  $T_{ab}^i$  and  $I_{ab}^i$  are respectively the nominal transform and the uncertainty term associated to the relative position of the objects  $A$  and  $B$  in  $W_i$ ; according to our model, the relation  $\prec$  is defined using the classical subset operator:  $I_{ab} \prec I_o \Leftrightarrow (Tr_{ab} \subset Tr_o) \wedge (U_{ab} \subset U_o) \wedge (D_{ab} \subset D_o)$ .

Then, a constrained manipulation plan will be considered as "correct", iff the constraints  $CO_i$ , for  $i = 0, 1 \dots n$ , are verified in the related world states  $W_i$  resulting from the execution of the actions  $A_1 A_2 \dots A_i$ .

### 3.5.2 Checking for the correctness of the plan

Let  $C_i$  be the conditions holding in  $W_i$  on the position variables – a condition is represented by a couple of the type  $(T_{ab} = t_o, I_{ab} = i_o)$ -. These conditions have been obtained after having “applied” the actions  $A_1 \dots A_i$ , to the initial conditions  $C_o$  holding in  $W_o$ . For instance, the uncertainty on the position of an object  $A$  in  $W_i$ , depends on both the inaccuracy of the used feeder and the errors introduced by the manipulation operations executed on  $A$ .

Then, computing  $C_i$  requires to propagate the initial conditions  $C_o$  through the actions  $A_1$  to  $A_i$ . Such a mechanism is referred as *forward propagation*. It leads to compute at each step  $k$  the *strongest postcondition*  $POST_k(C_{k-1})$ , obtained after having applied the action  $A_k$  to the conditions  $C_{k-1}$  holding in  $W_{k-1}$ . For example, the condition “ $I_{oa} = i_o$ ” will give rise to the condition “ $I_{oa} = i_o + I_{move}$ ”, after having executed the action “*MOVE A BY T*”. Such a computation is executed using the semantics of the actions as explained in section 3.4. It is recursively applied to the plan, using the following property:  $C_i = POST_i(C_{i-1})$ . Then, the plan will be considered as “correct” iff  $C_i$  implies  $CO_i$ , for  $i = 0, 1 \dots n$ .

**Remark:** A “correct” plan is guaranteed to work, provided that no unexpected physical fault occurs at execution time (an object fall for instance). But the method may sometimes lead to conclude that a quite reliable plan is not correct, because of the applied approximations.

### 3.5.3 Amending the plan

In case of failure (a constraint  $C$  is not verified in  $W_i$ ), the system applies a *backward propagation mechanism* for determining the uncertainty constraints which should hold in the previous world states, in order to guarantee that  $C$  will be verified in  $W_i$ . The main idea consists in computing at each step  $k$  the *weakest precondition*  $PRE_k(C_k)$  in  $W_{k-1}$ , which guarantee that the condition  $C_k$  will hold in  $W_k$ . For example, the condition “ $I_{oa} < i_o$ ” will hold after execution of the action “*MOVE A BY T*”, if  $I_{oa}$  verifies the condition “ $I_{oa} + I_{move} < i_o$ ” in the precedent world state. Such a computation is executed using the semantics of the actions as explained in section 3.4. It is recursively applied to the plan, using the property  $C_j^i = PRE_{j+1}(C_{j+1}^i) \wedge CO_j$ , where  $C_j^i$  represents the constraints of  $W_j$  ( $j < i$ ) which have been derived from  $CO_i$  using the backward propagation mechanism. Then, modifying  $W_j$  in order to verify the conditions  $C_j^i$ , leads to obtain a correct subsequence  $A_{j+1} \dots A_i$  (because each constraint in  $CO_k$  is verified, for  $k = j \dots i$ ).

The last step consists in determining *where* and *how* to amend the plan. Since any world state  $W_k$  (for  $k \leq i$ ) may be initially chosen, it is necessary to determine where the amending operation has to take place. Unfortunately, it seems that no decisive criterion exists for guiding this choice.

The other problem is to determine the amending strategy which seems to be the more appropriate. In our approach, this operation is executed by “*patching*” the assembly plan, using a well suited sensory based strategy. But, no practical method has currently been devised for solving this problem.

## 4 Conclusion

In this paper we have described two complementary methods for dealing with position and shape uncertainty when planning robot actions. The first method operates at a *local level* for planning the fine motion strategies required for achieving the strongly constrained assembly relations. The other method is applied in a second time. It operates at a *global level* for amending the produced manipulation plan, when some uncertainty constraints are not verified.

The fine motion planner described in the paper has been implemented in LUCID-LISP on a SUN 260. Most of the experimentations have been executed in simulation. Some of them have given rise to real executions using a six d.o.f SCEMI robot equipped with a force sensor. This is the case for

the example shown in figure 1 which has been successfully executed by the robot. 9 mn of CPU time was needed for synthesizing the related fine motion program. The constructed state graph was made of about 50 nodes and 80 arcs.

The verification/correction module has been partly implemented in LUCID-LISP on a SUN 260. The two related propagation mechanisms have been tested on some simple examples, using both the error bounds representation and a gaussian based model. Despite the theoretical differences existing between the two approaches, the obtained results look very similar when dealing with simple manipulation plans involving very few data fusion operations. However, more realistic experiments are still needed for really evaluating the scope of our method. A complementary work is also needed for solving the plan amendment problem.

### Acknowledgements:

This paper is based on the work done by Pascal Theveneau and Pierre Puget in the scope of the SHARP project of the LIFIA Laboratory. It was partly supported by the French National ARA project of the CNRS, the ADI agency, the ITMI company, and the INRIA institute.

### References

- [1] R.A.Brooks: "Symbolic error analysis and robot planning", International Journal of Robotics Research, vol 1, no 4, December 1982.
- [2] S.J.Buckley: "Planning and teaching compliant motion strategies", Ph.D Thesis, Artificial Intelligence Laboratory, MIT, January 1987.
- [3] R.Chatila, J.P.Laumond: "Position referencing and consistent world modeling for mobile robots", IEEE International Conference on Robotics and Automation, St.Louis, 1985.
- [4] B.Dufay, J.C.Latombe: "An approach to automatic robot programming based on inductive learning", 1st International Symposium on Robotics Research, Bretton Woods, August 1983.
- [5] H.F.Durrant-Whyte: "Consistant integration and propagation of disparate sensor observations", International Journal of Robotics Research, vol.6, nb.3, 1987.
- [6] M.Erdmann: "On motion planning with uncertainty", AI-TR-810, Artificial Intelligence Laboratory, MIT, 1984.
- [7] O.D.Faugeras, M.Hebert: "The representation, Recognition, and locating of 3D Objects", International Journal of Robotics Research, vol.5, nb.3, 1986.
- [8] C.Gandon: "Introduction de la compliance dans la programmation des robots", Thèse de 3ème Cycle, INPG, Grenoble, October 1986.
- [9] C.Laugier, P.Theveneau: "Planning sensor-based motions for part-mating using geometric reasoning", ECAI'86, Brighton, July 1986.
- [10] C.Laugier: "Raisonnement géométrique et méthodes de décision en robotique. Application à la programmation automatique des robots", Thèse d'Etat, INPG, Grenoble, December 1987.
- [11] C.Laugier: "Planning robot motions in the SHARP system", Published in "CAD based programming for sensory robots", Edited by Bahram Ravani, NATO ASI Series F, vol.50, Springer-Verlag, 1988.
- [12] T.Lozano-Perez: "The design of a mechanical assembly system", AI-TR-397, M.I.T. Artificial Intelligence Laboratory, Cambridge, December 1976.
- [13] T.Lozano-Perez, M.T.Mason, R.H.Taylor: "Automatic synthesis of fine-motions strategies for robots", 1st International Symposium of Robotics Research, Bretton Woods, August 1983.
- [14] T.Lozano-Perez, R.A.Brooks: "An approach to automatic robot programming", AI Memo 842, Artificial Intelligence Laboratory, M.I.T., April 1985.

- [15] M.T.Mason: "*Compliance and force control for computer controlled manipulators*", IEEE International Conference on Robotics and Automation, June 1981.
- [16] I.Mazon, P.Puget: "*Modélisation des incertitudes de positionnement*", Journées Géométrie et Robotique, Toulouse, May 1988.
- [17] P.Puget: "*Vérification-correction de programmes pour la prise en compte des incertitudes en programmation automatique des robots*", Thèse de l'Institut National Polytechnique de Grenoble, February 1989.
- [18] R.Smith, M.Self, P.Cheeseman: "*A stochastic map for uncertain spatial relationships*", IEEE International Conference on Robotics and Automation, Raleigh, March 1987.
- [19] R.H.Taylor: "*Synthesis of manipulator control programs from task-level specifications*", AIM 228, Stanford Artificial Intelligence Laboratory, July 1976.
- [20] P.Theveneau: "*Planification de mouvements fins de montage dans un système de programmation automatique de robots*", Thèse de l'Institut National Polytechnique de Grenoble, November 1988.
- [21] J.Troccaz, P.Puget: "*Dealing with uncertainties in robot planning using program proving techniques*", 4th International Symposium of Robotics Research, Santa Cruz, August 1987.
- [22] J.M.Valade: "*Raisonnement géométrique et synthèse de trajectoire d'assemblage*", Thèse de Docteur-Ingénieur, Université Paul Sabatier, Laboratoire d'Automatique et d'Analyse des Systèmes, Toulouse, January 1985.
- [23] D.E.Withney: "*Force feedback control of manipulator fine motions*", Journal of Dynamic Systems Measurement and Control, June 1977.



# Organising Geometric Computations for Space Telerobotics

Stephen Cameron

Robotics Group, Department of Engineering Science  
University of Oxford, Oxford OX1 3PJ, U.K.  
E-mail: stephen@uk.ac.oxford.robots

## Abstract

*A truly intelligent system that interacts with the physical world must be endowed with the ability to compute with shapes: despite this spatial reasoning is rarely regarded as part of mainstream A.I. We argue that the study of "intelligent" spatial algorithms is a worthwhile activity, and give opinions and suggestions for the way forward.*

## 1 What is Spatial Reasoning?

A truly intelligent system that interacts with the physical world must be endowed with the ability to compute with shapes. Despite this the study of "spatial reasoning" is a young field and is often confused with CAD/CAM. One problem is convincing people that the problem is hard: we liken it to vision, which is a problem that humans deal with easily, but has proved immensely difficult to solve on the computer. To see why spatial reasoning is of interest in the field of space telerobotics, consider the following scenarios:

1. An unmanned orbital maintenance robot is sent to replace a failed module in a communications satellite. The replacement is effected by a teleoperated manipulator, which is commanded to secure itself to the satellite, open an access door, bring out the old module, and replace it with the new one. Such teleoperations are commonplace in earth-bound activities which require the handling of hazardous materials; in the terrestrial case a human provides all the control of the manipulator via a wired teleoperator system. However, although it is technically feasible to provide the control channels to and from a human operator to an orbital vehicle, there is an unavoidable complication: the time lag introduced by the sluggishness of electromagnetic waves as a communication medium. A distance of only 15 000km introduces a delay of 0.1s, which is sufficient to make the manipulation task much harder. What is required here is the ability for the manipulator itself to deal with closing the low-level control loop: the manipulator take over the low-level actions such as "grip", "apply torque", and "open door". To effect these operations reliably requires a system that can reason over the geometric properties of its environment.
2. A Martian Rover vehicle is exploring a small part of the surface of the red planet. Although in constant communication with Earth it too suffers from the commands being sent from Earth being decided on the basis of information it sent some time ago—in this case, upward of half an hour. Even for such a vehicle to survive under such conditions it must at least be able to react to such immediate hazards as boulders or crevices in its path; to operate

efficiently it should be able to take some ‘sensible’ action in such cases (e.g., navigate around boulders). It is important to realise that the problem is not just one of recognising the hazard: once the hazard has been identified the appropriate corrective action depends crucially on its *geometry*.

3. A future space station is constructed by a number of astronauts with the help of a number of assembly robots. These robots are smarter versions of the maintenance vehicle (scenario 1) which are under the command of the astronauts, who use spoken commands made up from a small vocabulary to ease the data input problem. Thus these robots must be able to decide on how to parse crude assembly plans (as given by the astronauts) and produce detailed plans that they can execute. The parsing of these plans should take into account the relevance of the potential detailed plan to the overall mission (i.e., they should reject nonsensical parses) and, for reasons of safety, the robots should also try to foresee potential dangers to either the astronauts or the fabric of the space station. (Such a robot would probably require a good sense of dynamics and statics, as well as geometry.)

One thing that is clear from these examples is that all the tasks are “easy” for humans; we do them without conscious effort. However even the simplest of the tasks (scenario 1) would stretch our knowledge of how to write programs to deal with these problems. We first became interested in spatial problems through work in assembly robotics (e.g., [26,1]); however the problems occur in many guises. This paper explores the spectrum of such problems, and explains our first attempts to build systems that are truly spatially aware.

### 1.1 Types of Spatial Reasoning Problems

We can get a better grasp of the extent of spatial reasoning problems by listing some of the specific problems that have been tackled.

- *Storing the shape of objects.* This is an obvious, though non-trivial, prerequisite for most spatial reasoning problems. Storing shapes has been the subject of much research in the solid modelling community, though there the application is often to render pictures of objects.
- *Visual Recognition of Objects.* This problem has been studied for many years now, with limited, but steady, progress.
- *Interference Detection.* Given a number of objects in known positions and orientations, do they overlap?
- *Collision Detection.* Given a number of objects with known motions, do they collide?
- *Collision Avoidance.* Given a number of objects with start and goal positions, plan collision free motions.
- *Grasp Planning.* A specialised form of collision avoidance, where the goal is to obtain a stable grasp position.
- *Cloth Cutting.* Given a number of two-dimensional templates and a quantity of stock material, cut the items from the stock so as to minimise wastage.
- *Bin Packing.* Three dimensional analogue of cloth cutting, where we want to fit some parts into a bin. However if, say, the parts are electronic components which are to be connected

to form a device we would have further constraints on the solution, such as: minimising the wiring, allowing some parts to dissipate heat, and fixing controls near to the surface of the bin.

- *Pipe Routing.* Given an aircraft, plan routes for the various electrical ducts, hydraulic pipes, etc., while taking into account safety factors and other constraints.
- *Designing a Mechanical Assembly.* Here the input is a specification of the *function* of the assembly; the output should be a physical description of a suitable assembly (and preferably instructions as to how to make it).

It can be seen that spatial reasoning problems occur in a number of guises. The first problem (storing shapes) has been the subject of much research in the geometric modelling community, although it has often been with the main purpose of producing pictures of them. Some of the problems have reasonably well understood solutions; collision detection has been studied in robotics, vision systems are in practical use, and grasp planning is possible for some manipulators. Some of the problems have been well studied in theory, though the solutions so found are still to be incorporated in a practical system (collision avoidance), while others are only solved by humans with the help of a computer (cloth cutting, bin packing, pipe routing).

## 2 Problems with Spatial Reasoning

It should be clear that there are a large number of practical problems that require spatial reasoning for their solutions; we were interested in looking for patterns that would help to find general solutions. Examination of the spatial reasoning problems—some on paper, some using computers—have lead us to identify certain generic problems and trends.

- *No single method of solution.* When solutions are known to these problems several different methods can be identified, with no one method being general. (For example, [9] describes three different ways of performing collision detection, and [12] describes another.)
- *Different Representations.* There is no single obvious way to store a shape, and different methods of solving spatial reasoning problems may require different representations, and thus have different functionality<sup>1</sup>. This lack of a normal form for shape descriptions means that it is difficult to write algorithms that work in all cases (as Murphy's Law ensures that whichever shape representation system you use, somebody else will want to use a different one), and conversion between the existing shape descriptions is slow at best and an unsolved problem at worst.
- *Computer Arithmetic is of Finite Precision.* Although we can compute quantities to whatever precision we like, for most spatial quantities we cannot represent them exactly; this can cause many spatial algorithms to be ill-conditioned.
- *Shapes are Never Exact.* All of the common methods of shape representation denote idealised shapes; any real object has a shape which is never exactly known, but instead has been manufactured or measured to some tolerance (or combination of tolerances).

---

<sup>1</sup>The classic contenders for primary representation in the solid modelling community are Constructive Solid Geometry and the Boundary Representation, but others are known and are useful.

- *Objects never have random shapes.* Many implementations of low-level geometric algorithms work well with “random” shapes, but can have problems with shapes that show a pattern. Unfortunately many objects show such patterns<sup>2</sup>.

These problems might at first sight seem terminal, but we believe that they might be overcome by suitable *modularisation* of spatial subsystems. By using meta-level reasoning we might identify which particular spatial algorithm will perform a given task in the best manner. By hiding the representation details within a module we can write application programs that need not care which particular representation is used. By only demanding conservative answers to queries, and never “exact” answers, we can leave the lowest level modules the problem of deciding what sort of arithmetic is required, and what sorts of tolerances.

At present we have little hard evidence for the usefulness of this view. However we are currently implementing a system designed with this modular philosophy in mind, namely the spatial reasoning components for the Oxford Autonomous Guided Vehicle research.

### 3 The Oxford Autonomous Guided Vehicle Project

The Oxford Autonomous Guided Vehicle Project is a serious attempt to integrate many recent advances in sensing and spatial planning to provide a reliable system that can operate in a semi-structured (factory) environment. The is using a research version of the a vehicle developed and manufactured jointly by GEC plc (in the UK) and the Caterpillar Corporation (in the USA). This version has the same guidance, control and sensing capabilities of the standard vehicle, but it is smaller (more suitable for our laboratory) and is built to allow the fitting of extra equipment. To deal with the uncertainties in the world we must detect them, and so a number of different sensor systems are being attached to the vehicle, including vision cameras, a sonar array, a depth sensor, and an infra-red sensor. These sensor systems are major research projects in their own right (see [5] for an overview). The reason for having a number of different sensors is to be able to combine their output, with the noisy or poor data from one sensor being made up by better information from others. This is the domain of *sensor data fusion*, which is another major research topic. From the point of view of this paper the sensor data fusion system forms a convenient bridge between the sensor systems themselves, and the geometric models which define the *planning* systems’ model of the world. This role is highlighted by the overall system architecture of the project (figure 1).

Effectively information flows *from* the sensors to the sensor data fusion stage, and the sensor data fusion stage updates the world model with information it regards as pertinent and reliable. This architecture effectively allows the spatial planning systems to operate on the assumption that the information from the sensing systems is perfect; we are ignoring any tolerances in the data. (This is excusable in the application domain, but might not be, of course, in other domains.)

Two planning modules are shown connected to the world model. These correspond to two basic modes of the vehicles operation, namely motion between start and goal points whilst avoiding obstacles (as detected by the sensing systems), and acquisition of a pallet from a pile of mixed pallets and boxes. These operations form an important subset of the operations performed by typical factory vehicles. These modes were chosen for a particular reason: the first mode involves planning a path that avoids objects, whereas the second mode involves planning a path

---

<sup>2</sup>An example which we know well is the computation of configuration space obstacles by the vertex-set difference method [19]; this produces a regular set of points which broke most of the convex hull algorithms that we fed it to.

that contacts objects (*viz.*, picking up pallets using a fork-lift attachment). The support of these two modes within the *same* system is a major challenge for our modular approach. The final module—the overall planner—is a relatively simple task planner that selects a subtask using information supplied by a central factory computer. (Figure 2 shows a model of our vehicle approaching a pallet in our laboratory.)

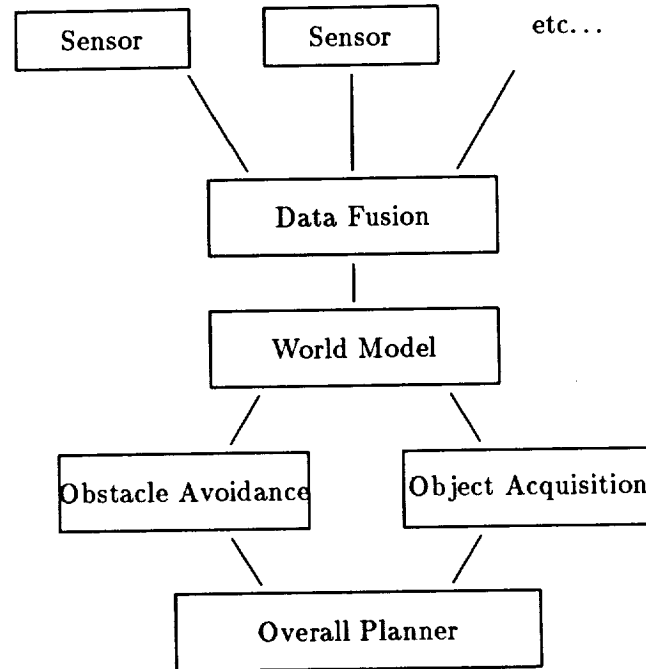


Figure 1: Overall System Architecture

In fact each of the three spatial reasoning modules (the world model, the path planner, and the acquisition module) are themselves modular, as discussed below.

### 3.1 The World Model

The world model accepts requests for information from the two spatial planning modules, and uses a combination of four internal models to answer the requests. This view of the world model is sketched in figure 3.

The four components of the world model can be divided into two pairs, consisting of static and dynamic information. The factory layout model “looks” like a two-dimensional plan of the factory, on which are marked static items (e.g., machining centres, pillars, doorways), quasi-static items (e.g., waste bins, doors), and nominal roadways. The 3D models are three-dimensional representations of objects that the vehicle senses or (literally) comes into contact with, for which a simplified two-dimensional projection will not suffice. (If there are many instances in the factory of, say, parts bins, only a single instance is stored in this component.) This split between two-dimensional and three-dimensional information is there partly because it lends itself to the makings of an efficient system, but mainly because it is natural: factory layouts are a common representation (and are good for planning routes), and three-dimensional databases are generally used for holding *instances* of objects. When the sensing systems require information

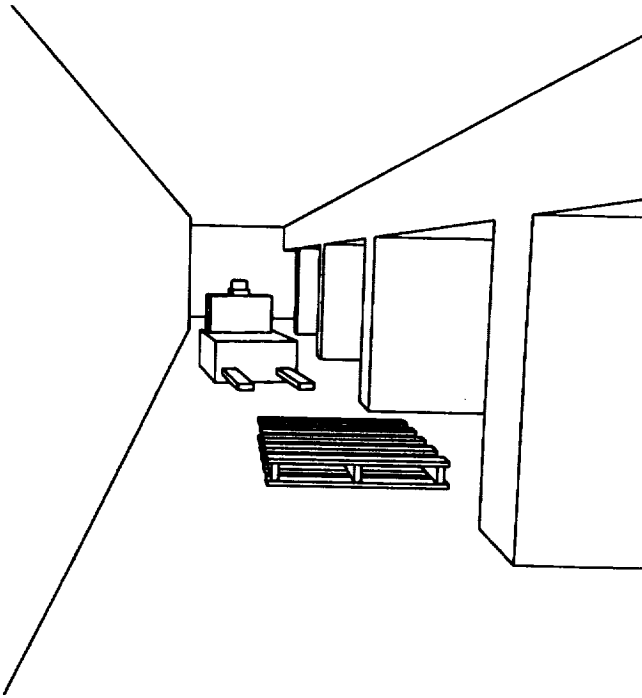


Figure 2: A ROBMOD Model of the AGV Laboratory

about what is visible, the kernel should refer to the layout model to discover which objects are (potentially) within the view, make up a three-dimensional model using instances from the solid modeller, and extract the visibility information from that.

The other components of the world model will change, both due to the discovery of unexpected objects and due to the movement of the vehicle itself.

### 3.2 Obstacle Avoidance

The fact that the environment of the AGV is reasonably well-structured means that we can take advantage of very simple path planning algorithms; in particular, much of the time the AGV can use generate-and-test, whereby a path is proposed and then checked for validity. In turn, proposing paths for the AGV is normally quite simple, as unless there are reasons to do otherwise the vehicle can just use the factory roadways. The only real problem occurs when an unexpected obstacle is encountered, when we expect one of three strategies to be used:

- If the obstacle is small we will use a potential-field approach to attempt to define a detour motion around it [16]; this motion is verified by the path-checker before being accepted.
- If the obstacle is larger the system will use a C-space approach, using a number of two-dimensional C-space maps covering a small number of vehicle orientations [19,18].
- If the route is blocked the vehicle will try to backtrack to find another route.

To perform collision detection we will use the routines already built into the ROBMOD system [10,9]. These routines have been optimised to perform intersection tests using S-bounds, which is a simple method for reasoning about the bounding volumes [8].

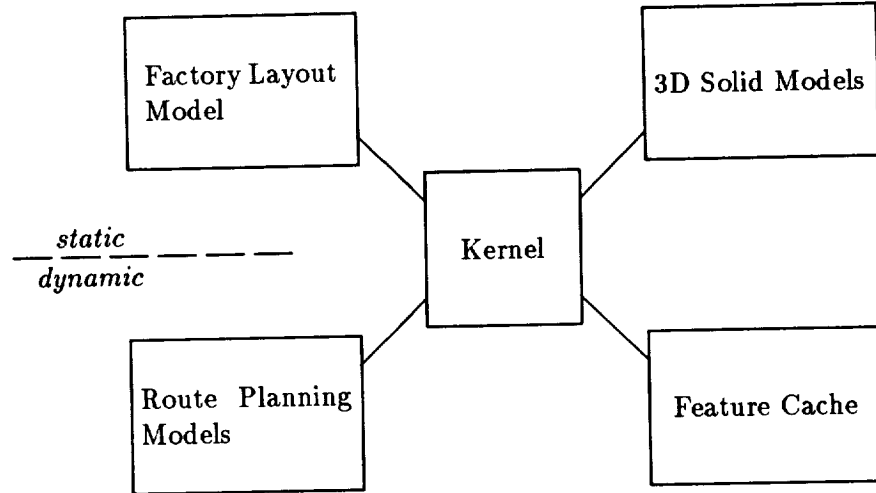


Figure 3: World Model Components

### 3.3 Object Acquisition

The purpose of the object acquisition experiment is to introduce the AGV into a space into which a number of loaded pallets have been positioned in an irregular manner. The AGV will have a fork-lift attachment, and has to identify the pallets, compute their orientations, and plan how the acquire the pallets using the fork-lift. In doing so it must take into account the positions of other objects and pallets in the area in order to avoid collisions. The path planning required in this case is thus of a different calibre from that required for obstacle avoidance, as it is necessary for the forks of the vehicle to come into close proximity with other objects. However, the class of objects that has to be tackled is restricted—namely, in the first instance, to pallets. Thus our approach is to use simple skeletonised plans to propose paths for the vehicle, which are then tested for validity. This will clearly work in simple cases; the challenge will come in getting the system to work well in relatively cluttered cases.

## 4 Related Work

Most of the push for what we now call geometric or solid modelling came from the engineering community [6,28,29]; [2] is an exception it that it was originally intended for vision research, although it was never really used for that purpose.

Collision avoidance has long been of interest in the robotics community [25,30]; more recently the configuration space approaches have been popular [21,19,33], although other methods are known [24,16]. No general methods for grasp planning have yet been developed [32,23]. Interference detection has been well-studied [3] and many methods for performing collision detection are known [10,12]. Mechanical assembly design is the study of the Design to Product project [14,27].

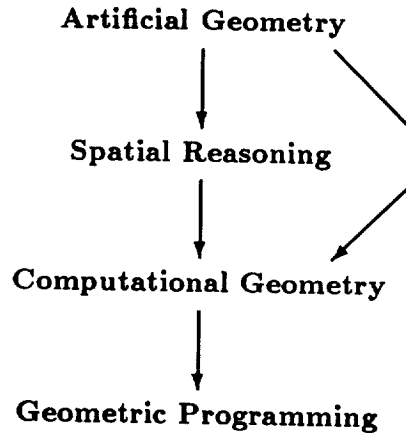


Figure 4: The World of Spatial Reasoning.

## 5 Looking Ahead

Although the implementation work for the AGV project is still in its infancy, we can see a potential problem ahead if we were to try to extend our strict, hierarchal model of spatial reasoning to more complex problems. Effectively, our current model has three levels. At the top level there is the spatial reasoning ‘module’, that given a well-posed problem selects a suitable algorithm to solve it. The next level down is the realm of computational geometry; the algorithms themselves. These algorithms have to be implemented on real computers, and so there is a final level where these algorithms are converted into suitable code—I call this the geometric programming level, and it is currently handled by humans. The potential problem is that exemplified by the fact that many computational geometry algorithms will break on certain inputs. If this happens with our strict hierarchy there is no mechanism to report useful information back. The eventual solution to this problem is unclear; I postulate an overseer level that sits on top of the existing hierarchy and can sense when things are going badly. With tongue in cheek I have dubbed this extra level the “Artificial Geometry” level (figure 4).

Taking the idea of an artificial geometry expert one stage further, we could envisage such an overseer that could write the computational algorithms for itself; at present only humans can do this task, which is more of an art than a science. Of course, we have yet very little idea as to how we could construct such an “expert”; however, we are keeping its future existence in mind as we tackle some of the very difficult problems on the lower levels of the spatial reasoning hierarchy.

## Acknowledgements

Many of the ideas in this paper took form whilst the author was employed under the McDonnell Douglas Independent Research and Development Programme. Karl Kempf and Bob Culley were responsible for encouraging my interest in spatial reasoning, and provided much of the inspiration (and much of the legwork) for this work. Also thanks to Mike Brady, and others in the Oxford Robotics Group, for feedback on these ideas. The AGV project is supported by the ACME Directorate of the SERC, and the author was supported by an SERC Atlas Research



Fellowship during the early stages of that programme. We owe thanks to our supporters on the AGV project, both from academia and elsewhere, and especially to GEC-FAST for providing us with the test vehicle.

## References

- [1] A. P. Ambler, S. A. Cameron, and D. F. Corner. Augmenting the RAPT robot language. In U. Rembold and K. Hormann, editors, *Languages for Sensor-Based Control in Robotics*, pages 305–316, Springer-Verlag, ref. F29 (1987), Castelvechio Pascoli, September 1986. Also as University of Edinburgh DAI Research Paper 330.
- [2] B. G. Baumgart. *Geometric Modelling for Computer Vision*. PhD thesis, Stanford, October 1974. Reprinted as Stanford AIM-249.
- [3] J. W. Boyse. Interference detection among solids and surfaces. *Communications of the ACM*, 22(1), 1979.
- [4] J. M. Brady. Artificial intelligence and robotics. *Artificial Intelligence*, 26(1):79–121, April 1985.
- [5] Michael Brady, Stephen Cameron, Hugh Durrant-Whyte, Margaret Fleck, David Forsyth, Alison Noble, and Ian Page. Progress towards a system that can acquire pallets and clean warehouses. In *Fourth Int. Symp. Robotics Research*, pages 359–374, Santa Cruz, August 1987.
- [6] I. C. Braid. *Designing with Volumes*. PhD thesis, University of Cambridge (U.K.), 1973.
- [7] R. A. Brooks. Symbolic error analysis and robot planning. *Int. J. Robotics Research*, 1(4):29–68, Winter 1982.
- [8] S. A. Cameron. Efficient intersection tests for objects defined constructively. *Int. J. Robotics Research*, 8(1), February 1989. Similar to Oxford Robotics Group OU-RRG-87-1.
- [9] S. A. Cameron. *Modelling Solids in Motion*. PhD thesis, University of Edinburgh, 1984. Available from the Department of Artificial Intelligence.
- [10] S. A. Cameron. A study of the clash detection problem in robotics. In *IEEE Int. Conf. Robotics and Automation*, pages 488–493, St. Louis, March 1985.
- [11] Stephen Cameron and Jon Aylett. ROBMOD: a geometry engine for robotics. In *IEEE Int. Conf. Robotics and Automation*, pages 880–885, Philadelphia, April 1988.
- [12] John Canny. On detecting collisions between moving polyhedra. *IEEE Pattern Analysis and Machine Intelligence*, 8(2):200–209, March 1986.
- [13] R. Chatila. Mobile robot navigation: space modelling and decisional processes. In *3rd Int. Sym. Rob. Res.*, Gouvieux, 1985.
- [14] Paul Fehrenbach and Tim Smithers. Design and sensor-based robotic assembly in the “design to product” project. In *Fourth Int. Symp. Robotics Research*, Santa Cruz, August 1987.
- [15] E. Kant. Understanding and automating algorithm design. In *9th IJCAI*, Los Angeles, 1985.
- [16] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *Int. Conf. Robotics and Automation*, pages 500–505, St. Louis, March 1985.
- [17] L. I. Liebermann and M. A. Wesley. AUTOPASS: an automatic programming system for computer-controlled mechanical assembly. *IBM Journal of Research and Development*, 21:321–333, July 1977.
- [18] T. Lozano-Pérez. Motion planning for simple robot manipulators. In *3rd Int. Sym. Rob. Res.*, Gouvieux, 1985.
- [19] T. Lozano-Pérez. Spatial planning—a configuration space approach. *IEEE Transactions on Computers*, 108–120, February 1983.

- [20] T. Lozano-Pérez, M. T. Mason, and R. H. Taylor. Automatic synthesis of fine-motion strategies for robots. In *First Int. Sym. Robotics Research*, MIT Press, 1984.
- [21] T. Lozano-Pérez and M. A. Wesley. An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM*, 22(10):560-570, October 1979.
- [22] J. Malik and T. O. Binford. Reasoning in time and space. In *8th IJCAI*, 1983.
- [23] M. T. Mason and R. C. Brost. Automatic grasp planning: an operational space approach. In *Oxford Workshop Geometric Reasoning*, 1986.
- [24] J. K. Myers. *A Supervisory Collision-Avoidance System for Robot Controllers*. Master's thesis, Carnegie-Mellon University, 1981.
- [25] D. L. Pieper. *The Kinematics of Manipulators under Computer Control*. PhD thesis, Stanford, 1968.
- [26] R. J. Popplestone, A. P. Ambler, and I. M. Bellos. RAPT: a language for describing assemblies. *Industrial Robot*, 5(3):131-137, 1978.
- [27] R. J. Popplestone, T. Smithers, J. Corney, A. Koutsou, K. Millington, and G. Sahar. Engineering design support systems. In *1st Int. Conf. Appl. AI Eng. Prob.*, Southampton, 1986.
- [28] A. A. G. Requicha. Representations for rigid solids: theory, methods and systems. *Computing Surveys*, 12(4), December 1980.
- [29] A. A. G. Requicha and H. B. Voelcker. Solid modeling: a historical summary and contemporary assessment. *IEEE Computer Graphics and Applications*, 2(2), March 1982.
- [30] S. Udupa. Collision detection and avoidance in computer controlled manipulators. In *Int. Joint Conf. Art. Int.*, Boston, 1977.
- [31] Wu Wen-tsun. Basic principles of mechanical theorem proving in geometries. *J. Sys. Sci. Math. Sci.*, 4(3), 1984.
- [32] M. P. Wingham. *Planning How to Grasp Objects in Cluttered Environments*. Master's thesis, Department of Artificial Intelligence, University of Edinburgh U.K., 1977.
- [33] C. K. Yap. Algorithmic motion planning. In J. T. Schwartz and C. K. Yap, editors, *Advances in Robotics 1*, Erlbaum, Hillsdale NJ, 1986.

# A Tesselated Probabilistic Representation for Spatial Robot Perception and Navigation

Alberto Elfes

Robotics Institute  
Carnegie Mellon University  
Pittsburgh, PA 15213

## Abstract

*The ability to recover robust spatial descriptions from sensory information and to efficiently utilize these descriptions in appropriate planning and problem-solving activities are crucial requirements for the development of more powerful robotic systems. Traditional approaches to sensor interpretation, with their emphasis on geometric models, are of limited use for autonomous mobile robots operating in and exploring unknown and unstructured environments. In this paper, we present a new approach to robot perception that addresses such scenarios using a probabilistic tessellated representation of spatial information called the Occupancy Grid. The Occupancy Grid is a multi-dimensional random field that maintains stochastic estimates of the occupancy state of each cell in the grid. The cell estimates are obtained by interpreting incoming range readings using probabilistic models that capture the uncertainty in the spatial information provided by the sensor. A Bayesian estimation procedure allows the incremental updating of the map using readings taken from several sensors over multiple points of view. We provide an overview of the Occupancy Grid framework and illustrate its application to a number of problems in mobile robot mapping and navigation. We argue that a number of robotic problem-solving activities can be performed directly on the Occupancy Grid representation, and draw some parallels between operations on Occupancy Grids and related image processing operations.*

## 1 Introduction

Two crucial requirements for the development of more flexible and powerful robotic systems are the ability to recover robust spatial descriptions of the surrounding world using sensory information, and the ability to efficiently utilize these descriptions in appropriate planning and problem-solving activities. Traditional approaches to sensor interpretation in Robotics and Computer Vision have largely relied on the recovery and manipulation of geometric world models [6]. "Low-level" sensing procedures extract geometric features such as line segments or surface patches from the sensor data, while "high-level" sensor modules use prior geometric models and heuristic assumptions about the environment to constrain the

sensor interpretation process. The resulting deterministic geometric descriptions of the environment of the robot are subsequently used as the basis for other robotic activities, such as obstacle avoidance, path-planning and navigation, or planning of grasping and assembly operations. These approaches, which we characterize as part of the *Geometric Paradigm* in Computer Vision, have, however, several shortcomings [6]. Generally speaking, the Geometric Paradigm leads to sparse and brittle world models; it requires early decisions in the interpretation of the sensor data for the instantiation of specific model primitives; it does not provide appropriate mechanisms for handling the uncertainty and errors intrinsic in the sensory information; and it relies heavily on the accurateness and adequacy of the prior world models and heuristic assumptions used. As a result, these geometric approaches are of limited use for complex scenarios such as those that arise in the use of autonomous or semi-autonomous vehicles for planetary exploration. Such mobile robots have to be able to operate in and explore unknown and unstructured environments, while coping with unforeseen conditions.

More recently, a number of other methodologies have started to be applied to robot perception tasks, with encouraging preliminary results. We have discussed elsewhere [6, 4] the role of stochastic sensor models and representation schemes in the development of robust robot systems operating in unstructured real-world environments.

In this paper, we review a new approach to robot perception and world modelling that uses a probabilistic tessellated representation of spatial information called the *Occupancy Grid* [6, 4]. The Occupancy Grid is a multi-dimensional random field that maintains stochastic estimates of the occupancy state of each cell in the grid. The cell estimates are obtained by interpreting incoming range readings using probabilistic models that capture the uncertainty in the spatial information provided by the sensors. Bayesian estimation procedures allow the incremental updating of the Occupancy Grid using readings taken from several sensors over multiple points of view. As a result, the disambiguation of sensor data is performed not through heuristics or prior models, but by higher sensing rates and the use of appropriate sensing strategies.

In subsequent sections, we will provide an overview of

the Occupancy Grid formulation and discuss how the Occupancy Grid framework provides a unified approach to a number of tasks in mobile robot perception and navigation. These tasks include range-based mapping, multiple sensor integration, path-planning and obstacle avoidance, handling of robot position uncertainty and other related problems. We suggest that a number of robotic problem-solving activities can be performed directly on the Occupancy Grid representation, precluding the need for the recovery of deterministic geometric descriptions. We also draw some parallels between operations on Occupancy Grids and related image processing operations.

## 2 The Occupancy Grid Approach

The scenario under consideration in this paper involves a mobile robot operating in unknown and unstructured environments, and carrying a complement of sensors that provide range information directly (sonar, scanning laser rangefinders) or indirectly (stereo systems). We will be mainly concerned with the development of robust mechanisms for robot perception and navigation. In this section, we provide a brief outline of the Occupancy Grid formulation, while in the succeeding sections we discuss several applications of Occupancy Grids to mobile robot mapping and navigation. More details can be found in [6, 4]; preliminary experimental results were reported in [8, 5, 9, 3, 13].

### 2.1 The Occupancy Grid Representation

The Occupancy Grid is a multi-dimensional (typically 2D or 3D) tessellation of space into cells, where each cell stores a probabilistic estimate of its state. Formally, an *Occupancy Field*  $O(x)$  can be defined as a discrete-state stochastic process defined over a set of continuous spatial coordinates  $x = (x_1, x_2, \dots)$ , while the *Occupancy Grid* is defined over a discrete spatial lattice. Consequently, the Occupancy Grid corresponds to a discrete-state (binary) random field [19]. A *realization* of the Occupancy Grid is obtained by estimating the state of each cell from sensor data.

More generally, the cell state could encompass a number of properties, described using a random vector associated with each lattice point of the random field, and estimated accordingly. We refer to such general world model representations, which are again instances of random fields, as *Inference Grids* [6]. Since in our current discussion we are mainly interested in *spatial* models for robot perception, we will restrict ourselves to the estimation of a single property, the *occupancy state* of each cell.

In the Occupancy Grid, the state variable  $s(C)$  associated to a cell  $C$  is defined as a discrete random variable with two states, *occupied* and *empty*, denoted OCC and EMP. Since the states are exclusive and exhaustive,  $P[s(C) = \text{OCC}] + P[s(C) = \text{EMP}] = 1$ . Each cell has, therefore, an associated probability mass function that is estimated by the sensing process.

### 2.2 Estimating the Occupancy Grid

To construct a map of the robot's environment, two processing stages are involved. First, a sensor range measurement  $r$  is interpreted using a stochastic sensor model. This model is defined by a probability density function (p.d.f.) of the form  $p(r | z)$ , where  $z$  is the actual distance to the object being detected. Secondly, the sensor reading is used in the updating of the cell state estimates of the Occupancy Grid. For simplicity, we will derive the interpretation and updating steps for an Occupancy Grid defined over a single spatial coordinate, and outline the generalization to more dimensions.

In the continuous case, the random field  $O(x)$  is described by a probability mass function defined for every  $x$  and is written as  $O(x) = P[s(x) = \text{OCC}](x)$ , the probability of the state of  $x$  being *occupied*. The probability of  $x$  being *empty* is obviously given by  $P[s(x) = \text{EMP}](x) = 1 - P[s(x) = \text{OCC}](x)$ . The conditional probability of the state of  $x$  being occupied given a sensor reading  $r$  will be written as  $O(x | r) = P[s(x) = \text{OCC} | r](x)$ . For the discrete case, the Occupancy Grid corresponds to a sampling of the random field over a spatial lattice. We will represent the probability of a cell  $C_i$  being occupied as  $O(C_i) = P[s(C_i) = \text{OCC}](C_i)$ , and the conditional probability given a sensor reading  $r$  as  $O(C_i | r) = P[s(C_i) = \text{OCC} | r](C_i)$ . When only a single cell  $C_i$  is being referenced, we will use the more succinct notation  $P[s(C_i) = \text{OCC}]$ .

We now consider a range sensor characterized by a sensor model defined by the p.d.f.  $p(r | z)$ , which relates the reading  $r$  to the true parameter space range value  $z$ . Determining an optimal estimate  $\hat{z}$  for the parameter  $z$  is a straightforward estimation step, and can be done using Bayes' formula and MAP estimates [2, 18]. Recovering a model of the environment as a whole, however, leads to a more complex estimation problem. In general, obtaining an optimal estimate of the occupancy grid  $O(C_i | r)$  would require determining the conditional probabilities of all possible world configurations. For the two-dimensional case of a map with  $m \times m$  cells, a total of  $2^{m^2}$  alternatives are possible, leading to a non-trivial estimation problem. To avoid this combinatorial explosion of grid configurations, the cell states are estimated as *independent* random variables. This is equivalent to assuming that the Occupancy Grid is a Markov Random Field (MRF) of order 0 [19], and can be relaxed using estimation procedures for higher order MRFs [10, 12].

To determine how a sensor reading is used in estimating the state of the cells of the Occupancy Grid, we start by applying Bayes' theorem to a single cell  $C_i$ :

$$P[s(C_i) = \text{OCC} | r] = \frac{p[r | s(C_i) = \text{OCC}] P[s(C_i) = \text{OCC}]}{\sum_{s(C_i)} p[r | s(C_i)] P[s(C_i)]} \quad (1)$$

Notice that the  $p[r | s(C_i)]$  terms that are required in this equation do not correspond directly to the sensor model

$p(r | z)$ , since the latter implicitly relates the range reading to the detection of a single object surface. In other words, the sensor model can be rewritten as:

$$p(r | z) = p[r | s(C_i) = \text{OCC} \wedge s(C_k) = \text{EMP}, k < i] \quad (2)$$

To derive the distributions for  $p[r | s(C_i)]$ , it is necessary to perform an estimation step over all possible world configurations. This can be done using Kolmogoroff's theorem [15]:

$$p[r | s(C_i) = \text{OCC}] = \sum_{\{G_{s(C_i)}^O\}} (p[r | s(C_i) = \text{OCC}, G_{s(C_i)}^O] \times P[G_{s(C_i)}^O | s(C_i) = \text{OCC}]) \quad (3)$$

where  $G_{s(C_i)}^O = (s(C_1) = s_1, \dots, s(C_{i-1}) = s_{i-1}, s(C_{i+1}) = s_{i+1}, \dots, s(C_n) = s_n)$  stands for a specific grid configuration with  $s(C_i) = \text{OCC}$ , and  $\{G_{s(C_i)}^O\}$  represents all possible grid configurations under that constraint. In the same manner,  $p[r | s(C_i) = \text{EMP}]$  can be computed as:

$$p[r | s(C_i) = \text{EMP}] = \sum_{\{G_{s(C_i)}^E\}} (p[r | s(C_i) = \text{EMP}, G_{s(C_i)}^E] \times P[G_{s(C_i)}^E | s(C_i) = \text{EMP}]) \quad (4)$$

where  $G_{s(C_i)}^E$  is defined in a manner similar to  $G_{s(C_i)}^O$ , above.

The configuration probabilities  $P[G_{s(C_i)} | s(C_i)]$  are determined from the individual prior cell state probabilities. These, in turn, can be obtained from experimental measurements for the areas of interest, or derived from other considerations about likelihoods of cell states. We have opted for the use of non-informative or maximum entropy priors [1], which in this case reduce to equal probability assignments for the two possible states:

$$P[s(C_i) = \text{OCC}] = P[s(C_i) = \text{EMP}] = 1/2 \quad (5)$$

Finally, Eq. 2 is used in the computation of the distributions  $p[r | s(C_i)]$ . The full derivation of these terms is found in [6]; we only remark that because there are subsets of configurations that are *indistinguishable* under a single sensor observation  $r$ , it is possible to derive closed form solutions of these equations for certain sensor models, and to compute numerical solutions in other cases.

To illustrate the approach, consider the case of an ideal sensor, characterized by the p.d.f.  $p(r | z) = \delta(r - z)$ , where  $\delta$  is the Kronecker delta. For this case, the following closed form solution of Eq. 1 results (Fig. 1):

$$P[s(C_i) = \text{OCC} | r] = \begin{cases} 0 & \text{for } x < r, x \in C_i \\ 1 & \text{for } x, r \in C_i \\ 1/2 & \text{for } x > r, x \in C_i \end{cases} \quad (6)$$

which is an intuitively appealing result: if an ideal sensor measures a range value  $r$ , the corresponding cell has occupancy probability 1; the preceding cells are empty and have occupancy probability 0; and the succeeding cells have not been observed and are therefore unknown, having occupancy probability 1/2.

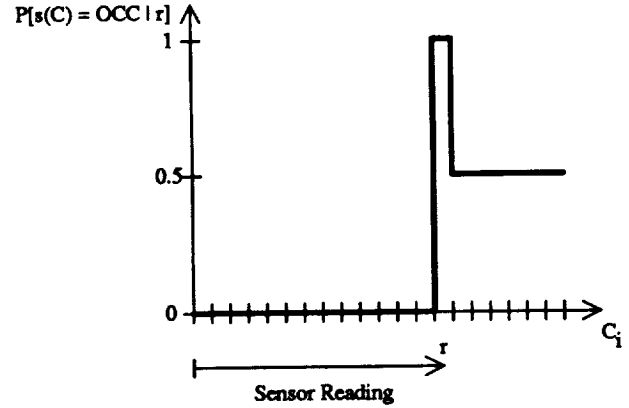


Figure 1: Occupancy Probability Profile for an ideal sensor, given a range measurement  $r$ .

As another example, consider a range sensor whose measurements are corrupted by Gaussian noise of zero mean and variance  $\sigma^2$ . The corresponding sensor p.d.f. is given by:

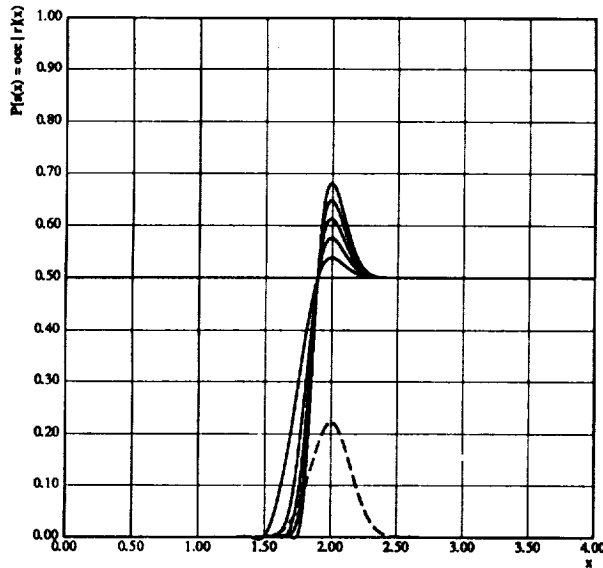
$$p(r | z) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(r - z)^2}{2\sigma^2}\right) \quad (7)$$

This equation can be used in the numerical evaluation of Eqs. 3 and 4. A plot of a typical cell occupancy profile obtained for this sensor from Eq. 1 is shown in Fig. 2.

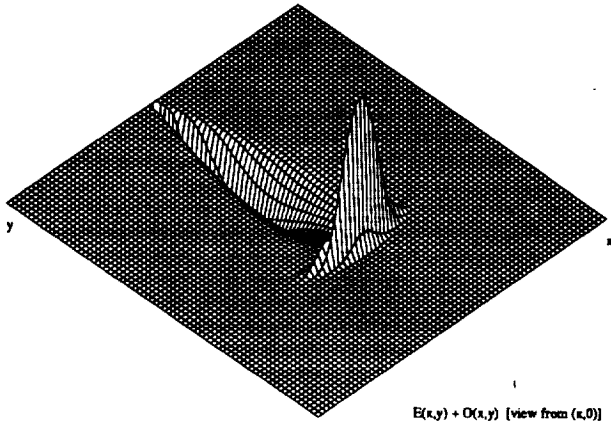
To extend the derivation to two spatial dimensions, consider the example of a range sensor characterized by Gaussian uncertainty in both range and angle, given by the variances  $\sigma_r^2$  and  $\sigma_\theta^2$ . In this case, the sensor p.d.f. can be represented in polar coordinates as:

$$p(r | z, \theta) = \frac{1}{2\pi\sigma_r\sigma_\theta} \exp\left[-\frac{1}{2}\left(\frac{(r - z)^2}{\sigma_r^2} + \frac{\theta^2}{\sigma_\theta^2}\right)\right] \quad (8)$$

In this formula, the dependency of the random variable  $r$  on  $z$  and  $\theta$  is decoupled, a reasonable assumption for a first-order model of certain types of range sensors. Consequently, the estimation of the two-dimensional Occupancy Grid can be performed conveniently in polar coordinates  $(\rho, \varphi)$ , using fundamentally the same formulation as above (Eqs. 3 and 4) and applying Eq. 8 to recover the distributions  $p[r | s(C_{\rho_i\varphi_j})]$ . These in turn are used to obtain the polar Occupancy Grid  $P[s(C_{\rho_i\varphi_j}) | r]$ . To generate the corresponding two-dimensional cartesian Occupancy Grid, the polar grid can be scanned and resampled. The results are similar to the 2D cartesian Occupancy Grid shown in Fig. 3, obtained from a single sonar reading. Similar derivations can be performed for 3D Occupancy Grids.



**Figure 2:** Occupancy Probability Profiles obtained from a sensor with Gaussian distribution. The sensor model  $p(r | z)$  is shown superimposed (dashed line). Several successive updates of the cell occupancy probabilities are plotted, with the sensor positioned at  $x = 0.0$  and with  $r = 2.0$ . The grid was initialized with  $P[s(x) = \text{OCC}] = 0.5$ . The profiles show that the Occupancy Grid converges towards the behaviour of the ideal sensor.



**Figure 3:** Two-Dimensional Sonar Occupancy Grid. The occupancy profile shown corresponds to a range measurement taken by a sonar sensor positioned at the upper left, pointing to the lower right. The plane shows the UNKNOWN level.

### 2.3 Updating the Occupancy Grid

Due to the intrinsic limitations of sensor systems, recovering a description of the world from sensory informa-

tion is fundamentally an underconstrained problem. As mentioned previously, this has historically been addressed by the heavy use of prior models and simplifying heuristic assumptions about the robot's environment. Within the Occupancy Grid framework, this problem is handled instead by the use of additional sensing to resolve sensor ambiguity and uncertainty. Rather than relying on a single observation to obtain an estimate of the Occupancy Grid, information from multiple sensor readings taken from different viewpoints is composed to improve the sensor-derived map. This leads naturally to an emphasis on higher sensing rates and on the development of adequate sensing strategies.

To allow the incremental composition of sensory information, we use the sequential updating formulation of Bayes' theorem [6]. Given the current estimate of the state of a cell  $s(C_i)$ ,  $P[s(C_i) = \text{OCC} | \{r\}_i]$ , based on observations  $\{r\}_i = \{r_1, \dots, r_i\}$ , and given a new observation  $r_{i+1}$ , we can write:

$$\begin{aligned} P[s(C_i) = \text{OCC} | \{r\}_{i+1}] &= \\ &= \frac{p[r_{i+1} | s(C_i) = \text{OCC}] P[s(C_i) = \text{OCC} | \{r\}_i]}{\sum_{s(C_i)} p[r_{i+1} | s(C_i)] P[s(C_i) | \{r\}_i]} \end{aligned} \quad (9)$$

In this formula, the previous estimate of the cell state,  $P[s(C_i) = \text{OCC} | \{r\}_i]$ , serves as the prior and is obtained directly from the Occupancy Grid. Tables for the sensor model-derived terms  $p[r_{i+1} | s(C_i)]$  can be computed offline and used in the updating procedure. The new cell state estimate  $P[s(C_i) = \text{OCC} | \{r\}_{i+1}]$  is subsequently stored again in the map. An example of this Bayesian updating procedure is shown in Fig. 2.

### 2.4 Sensor Integration

To increase the capabilities and the performance of robotic systems in general, a variety of sensing devices are necessary to support the different kinds of tasks to be performed. This is particularly important for mobile robots, where multiple sensor systems can provide higher levels of fault-tolerance and safety. Additionally, qualitatively different sensors have different operational characteristics and failure modes, and can therefore complement each other.

Within the Occupancy Grid framework, sensor integration can be performed using a formula similar to Eq. 9 for the combination of estimates provided by different sensors [6]. This allows the updating of the *same* Occupancy Grid by multiple sensors operating independently. Consider two independent sensors  $S_1$  and  $S_2$ , characterized by sensor models  $p_1(r | z)$  and  $p_2(r | z)$ . In this case, the integration of readings  $r_{S_1}$  and  $r_{S_2}$ , measured by sensors  $S_1$  and  $S_2$ , respectively, can be done using:

$$P[s(C_i) = \text{OCC} | r_{S_1}, r_{S_2}] =$$

$$= \frac{p[r_{s_2} | s(C_i) = \text{OCC}] P[s(C_i) = \text{OCC} | r_{s_1}]}{\sum_{s(C_i)} p[r_{s_2} | s(C_i)] P[s(C_i) | r_{s_1}]} \quad (10)$$

A different estimation problem occurs when separate Occupancy Grids are maintained for each sensor system, and integration of these sensor maps is performed at a later stage by composing the corresponding cell probability estimates. This requires the combination of probabilistic evidence from different sources [1]. Consider the two cell occupancy probabilities  $P_1 = P_{S_1}[s(C_i) = \text{OCC} | \{r\}_1]$  and  $P_2 = P_{S_2}[s(C_i) = \text{OCC} | \{r\}_2]$ , obtained from separate Occupancy Grids built using sensors  $S_1$  and  $S_2$ . The general solution to this problem involves the use of a *Superbayesian* approach [1]. It requires the definition of probabilistic models of the form  $f_{S_i}(P_{S_i}[s(C_i)] | s(C_i))$  for each sensor, which serve to provide an evaluation of the sensor performance. It can be shown [6] that for simple linear models, the Superbayesian estimation procedure is reduced to a probabilistic evidence combination method known as the *Independent Opinion Pool* [1]. This method, when applied to the combination of the two sensor-derived estimates,  $P_1$  and  $P_2$ , yields the simple formula [6]:

$$P[s(C_i) = \text{OCC} | P_1, P_2] = \frac{P_1 P_2}{P_1 P_2 + (1 - P_1)(1 - P_2)} \quad (11)$$

Though this method is suboptimal in a Bayesian sense, it provides a computationally simple updating procedure. In previous work, described in [9, 13], the Independent Opinion Pool approach was used to integrate Occupancy Grids derived separately from two sensor systems, a sonar array and a single-scanline stereo module, mounted on a mobile robot. An example of the resulting maps is presented in Section 3.2.

### 2.5 Incorporation of User-Provided Maps

Throughout this paper we are mainly concerned with scenarios where the robot is operating in unknown environments, so that no prior maps can be used. There are other contexts, however, where such information is available. For example, mobile robots operating inside nuclear facilities could access detailed and substantially accurate maps derived from blueprints, while planetary rovers could take advantage of global terrain maps obtained from orbiting platforms. Such information can be represented using symbolic and geometric models such as those described in [11]. The incorporation of these high-level user-provided maps can be done within the Occupancy Grid framework using the same methodology outlined in the previous sections. To provide a common representation, the geometric maps are scan-converted into an Occupancy Grid, with occupied and empty areas being assigned the corresponding probabilities. These user maps can subsequently be used as priors for sensor maps, or can be treated simply as another source of information to be integrated with sensor-derived maps.

### 2.6 Decision-Making

In certain contexts, it may be necessary to make discrete choices concerning the state of a cell  $C$ . For that, the optimal estimate is provided by the *maximum a posteriori* (MAP) decision rule [2], which can be written in terms of occupancy probabilities as:

$$\begin{cases} C \text{ is OCCUPIED} & \text{if } P(s(C) = \text{OCC}) > P(s(C) = \text{EMP}) \\ C \text{ is EMPTY} & \text{if } P(s(C) = \text{OCC}) < P(s(C) = \text{EMP}) \\ C \text{ is UNKNOWN} & \text{if } P(s(C) = \text{OCC}) = P(s(C) = \text{EMP}) \end{cases} \quad (12)$$

Additional factors, such as the cost involved in making different choices, can be taken into account by using other decision criteria, such as minimum-cost estimates [18]. Depending on the specific application, it may also be of interest to define an UNKNOWN band, as opposed to a single thresholding value. As argued in [6], however, many robotic tasks can be performed directly on the Occupancy Grid, obviating the need to make discrete choices concerning the state of individual cells. In path-planning, for example, the cost of a path can be defined by a risk factor directly related to the corresponding cell probabilities [8].

## 3 Using Occupancy Grids for Mobile Robot Mapping

We will now proceed to illustrate the Occupancy Grid approach by discussing some applications of Occupancy Grids to autonomous mobile robots. In this section, we summarize the use of Occupancy Grids in sensor-based mobile robot *Mapping*, while in Section 4 we provide an overview of the use of Occupancy Grids in mobile robot *Navigation*. The experimental results shown here have been mostly obtained in operating environments that can be adequately described by two-dimensional maps. We have recently started to extend our work to the generation and manipulation of 3D Occupancy Grids.

One possible flow of processing for sensor-derived mobile robot mapping applications is outlined below and summarized in Fig. 4. As the mobile robot explores and maps its environment, the incoming sensor readings are interpreted using probabilistic sensor models. The map of the world that the robot acquires from a single sensor reading is called a *Sensor View*. Various Sensor Views taken from a single robot position can be composed into *Local Sensor Maps*, which can be maintained separately for each sensor type. A composite description of the robot's surroundings is obtained through sensor integration of separate Local Sensor Maps into a *Robot View* (as mentioned previously, Robot Views can be generated directly from the different sensors). As a result, the Robot View encapsulates the information recovered at a single mapping location. As the robot explores its surroundings, Robot Views taken from multiple data-gathering positions are composed into a *Global Map* of the environment. This requires relative registration of the Robot Views, an issue that is addressed in Section 4.

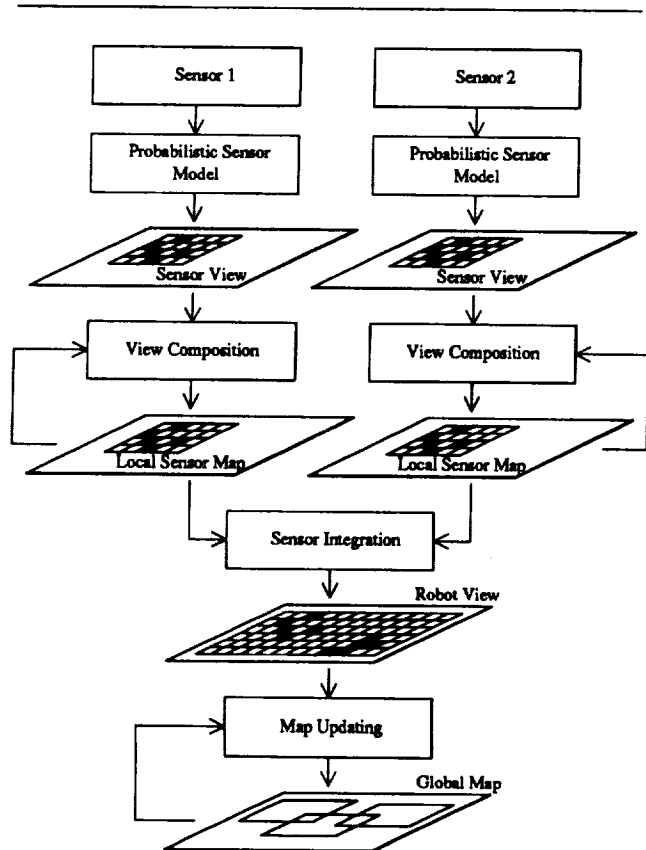


Figure 4: A Framework for Occupancy Grid Based Robot Mapping.

### 3.1 Sonar-Based Mapping

The Occupancy Grid representation was first developed in the context of sonar-based mapping experiments [14, 8]. The specific limitations of sonar sensors and the desire to recover robust and dense maps of the robot's environment precluded simple geometric interpretation methods [8] and led to the investigation of tessellated probabilistic representations. We developed an experimental system for sonar-based mapping and navigation for autonomous mobile robots called *Dolphin* [7, 8], and performed a number of indoor and outdoor experiments [6]. Fig. 5 shows a sonar map obtained during navigation down a corridor. Preliminary results were encouraging: the resulting sonar maps were robust and very useful for navigation. The cell updating mechanisms are computationally fast, allowing a high sensing to computation ratio. This led us to develop the Occupancy Grid formulation further and to apply it to other domains [6, 9, 13, 4].

### 3.2 Sensor Integration of Sonar and Scanline Stereo

The Occupancy Grid framework provides a straightforward approach to sensor integration. Range measurements from each sensor are converted directly to the Occupancy

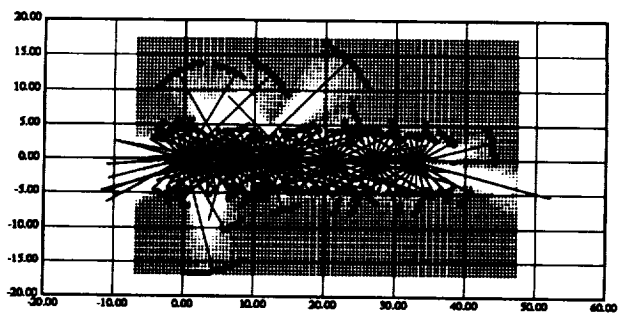


Figure 5: Sonar Mapping and Navigation Along a Corridor. Walls and open doors can be distinguished and enough resolution is present that even wall niches can be noticed in the map. The range readings taken from each robot stop are drawn superimposed on the map.

Grid representation, where data taken from multiple views and different sensors can be combined naturally. Sensors are treated modularly, and separate sensor maps can be maintained concomitantly with integrated maps, allowing independent or joint sensor operation. We have performed experiments in the integration of data from two sensor systems: a *sonar sensor array* and a *single-scanline stereo module* that provides horizontal depth profiles, both mounted on a mobile robot. This allows the generation of improved maps, taking advantage of the complementarity of the sensors [9, 13]. A typical set of maps is shown in Fig. 6.

## 4 Using Occupancy Grids for Robot Navigation

For autonomous robot navigation, a number of concerns have to be addressed. In this section, we briefly outline the use of Occupancy Grids in path-planning and obstacle avoidance, estimating and updating the robot position, and incorporating the positional uncertainty of the robot into the mapping process (Fig. 7).

### 4.1 Path-Planning and Obstacle Avoidance

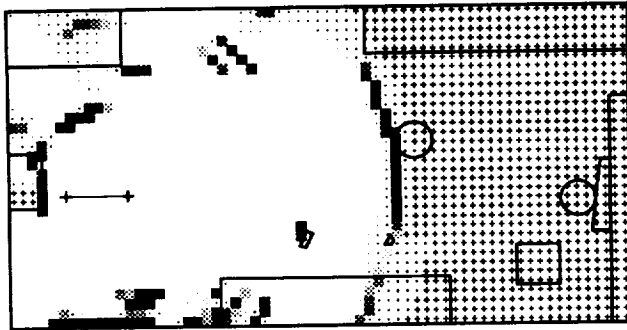
In the *Dolphin* system, path-planning and obstacle avoidance are performed using potential functions and an A\* search algorithm. The latter operates directly on the Occupancy Grid, optimizing a path cost function that takes into account both the distance to the goal and the occupancy probabilities of the cells being traversed [8, 6].

### 4.2 Handling Robot Position Uncertainty

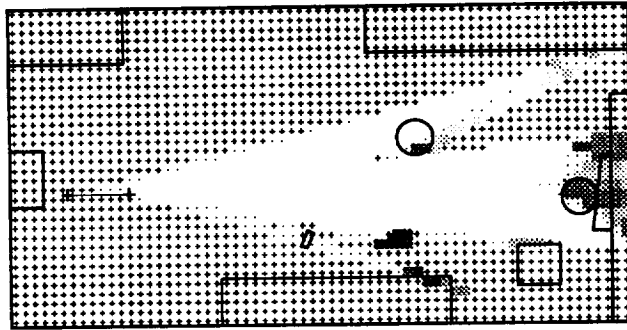
To desambiguate sensor information and recover accurate and complete descriptions of the environment of operation of a robot, it is necessary to integrate sensor data acquired from multiple viewing positions. To allow the



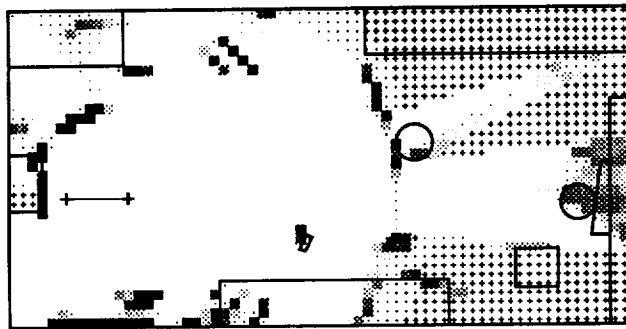
**Sonar Map:**



**Scanline Stereo Map:**

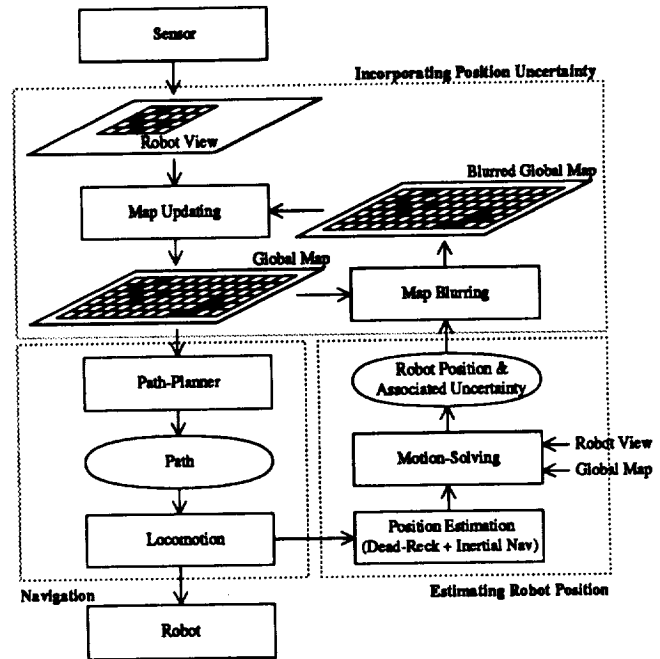


**Integrated Sonar and Scanline Stereo Map:**



**Figure 6: Sensor Integration of Sonar and Scanline Stereo.** Occupancy Grids generated separately for sonar and scanline stereo, and jointly through sensor integration are shown. *Occupied* regions are marked by shaded squares, *empty* areas by dots fading to white space, and *unknown* spaces by + signs.

composition of these multiple views into a coherent model of the world, accurate information concerning the relative transformations between data-gathering positions is necessary to allow precise registration of the views for subsequent integration. For mobile robots that move around in unstructured environments, recovering precise position information poses major problems. Over longer distances, dead-reckoning estimates are not sufficiently reli-



**Figure 7: A Framework for Occupancy Grid-Based Robot Navigation.**

able; consequently, motion-solving methods that use landmark tracking or map matching approaches are usually applied to reduce the registration imprecision due to motion. Additionally, the positional error is compounded over sequences of movements as the robot traverses its environment. This leads to the need for explicitly handling positional uncertainty and taking it into account when composing sensor information.

To represent and estimate the robot position as the vehicle explores its environment, we use the *Approximate Transformation* (AT) framework [16]. A robot motion  $M$ , defined with respect to some coordinate frame, is represented as  $\tilde{M} = \langle \hat{M}, \Sigma_M \rangle$ , where  $\hat{M}$  is the estimated (nominal) position, and  $\Sigma_M$  is the associated covariance matrix that captures the positional uncertainty. The parameters of the robot motion are determined from dead-reckoning and inertial navigation estimates, which can be composed using the AT *merging* operation, while the updating of the robot position uncertainty over several moves is done using the AT *composition* operation [16].

### 4.3 Motion-Solving

For more precise position estimation, a multi-resolution correlation-based motion-solving procedure is employed. It searches for an optimal registration between the new Robot View and the current Global Map, by matching the corresponding Occupancy Grids before map composition [14].

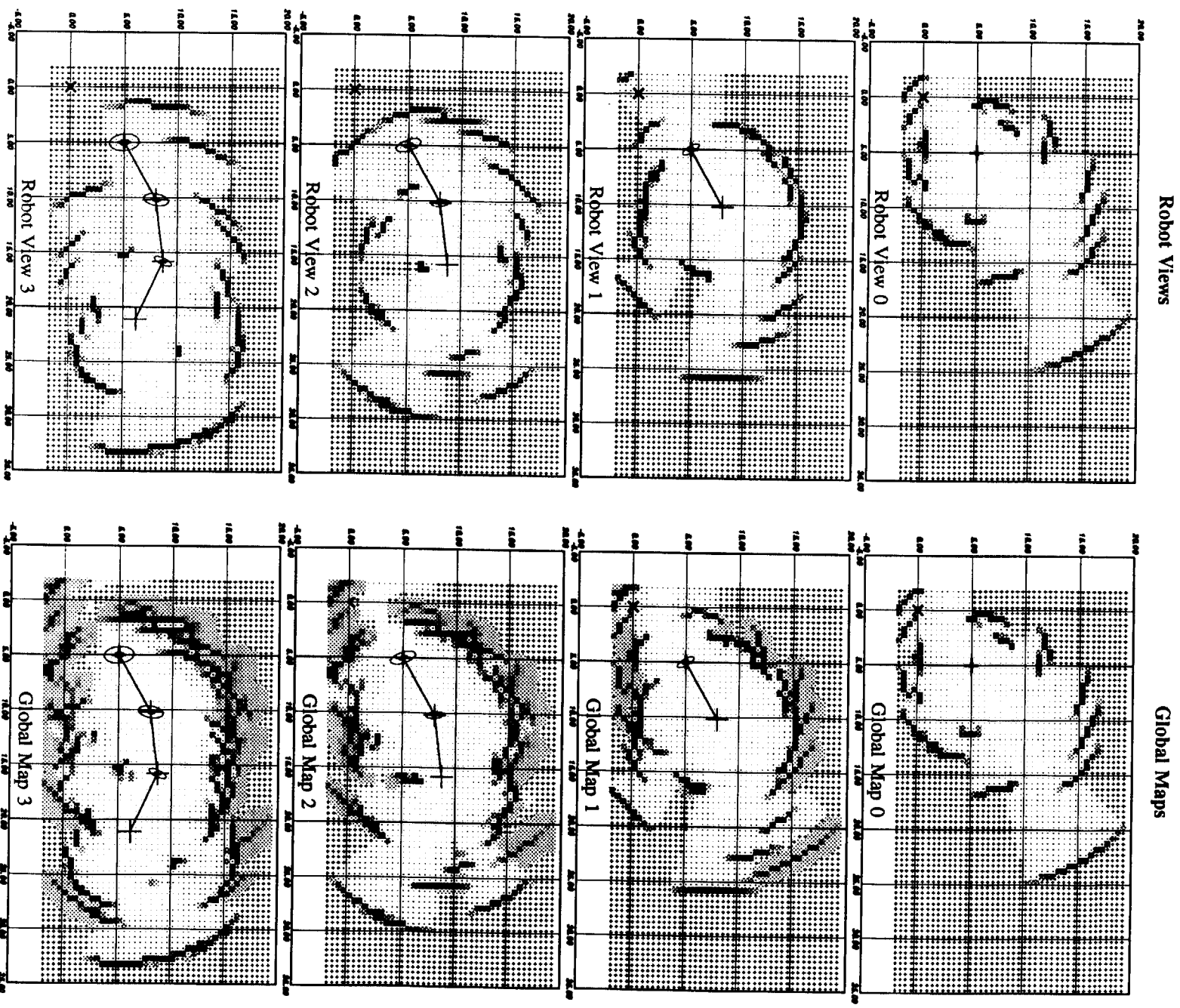


Figure 8: Incorporating Motion Uncertainty into the Mapping Process. For robot-centered mapping, the Global Map is blurred by the robot position uncertainty (shown using the corresponding covariance ellipses) prior to composition with the Robot View.

#### 4.4 Incorporating Positional Uncertainty into the Mapping Process

After estimating the registration between the new Robot View and the Global Map, the associated uncertainty is incorporated into the map updating process as a blurring or convolution operation performed on the Occupancy Grid. We distinguish between *World-Based Mapping* and *Robot-Based Mapping* [6, 4].

In *World-Based Mapping*, the motion of the robot is related to the observer or world coordinate frame, and the current Robot View is blurred by the robot's positional uncertainty prior to composition with the Global Map. If we represent the Global Map by  $M_G$ , the current Robot View by  $V_R$ , the robot position by the AT  $\tilde{R} = \langle \tilde{R}, \Sigma_R \rangle$ , the blurring operation by the symbol  $\tilde{\otimes}$  and the composition of maps by the symbol  $\tilde{\oplus}$ , we can express the world-based mapping procedure as:

$$M_G \leftarrow M_G \tilde{\oplus} (V_R \tilde{\otimes} \tilde{R}) \quad (13)$$

Since the global robot position uncertainty increases with every move, the effect of this updating procedure is that the new Views become progressively more blurred, adding less and less useful information to the Global Map. Observations seen at the beginning of the exploration are "sharp", while recent observations are "fuzzy". From the point of view of the inertial observer, the robot eventually "dissolves" in a cloud of probabilistic smoke.

For *Robot-Based Mapping* (Fig. 8), the registration uncertainty of the Global Map due to the recent movement of the robot is estimated, and the Global Map is blurred by this uncertainty prior to composition with the current Robot View. This mapping procedure can be expressed as:

$$M_G \leftarrow V_R \tilde{\oplus} (M_G \tilde{\otimes} \tilde{R}) \quad (14)$$

A consequence of this method is that observations performed in the remote past become increasingly uncertain, while recent observations have suffered little blurring. From the point of view of the robot, the immediate surroundings (which are of relevance to its current navigational tasks) are "sharp". The robot is leaving, so to speak, an expanding "probabilistic trail" of weakening observations behind it (see Fig. 8).

It should be noted, however, that the local spatial relationships observed within a Robot View still hold. So as not to lose this information, we use a two-level spatial representation, incorporating Occupancy Grids and Approximate Transformations. On one level, the individual Views are stored attached to the nodes of an AT graph (a *stochastic map* [17]) that describes the movements of the robot. Coupled to this, a Global Map is maintained that represents the robot's current overall knowledge of the world (Fig. 9).

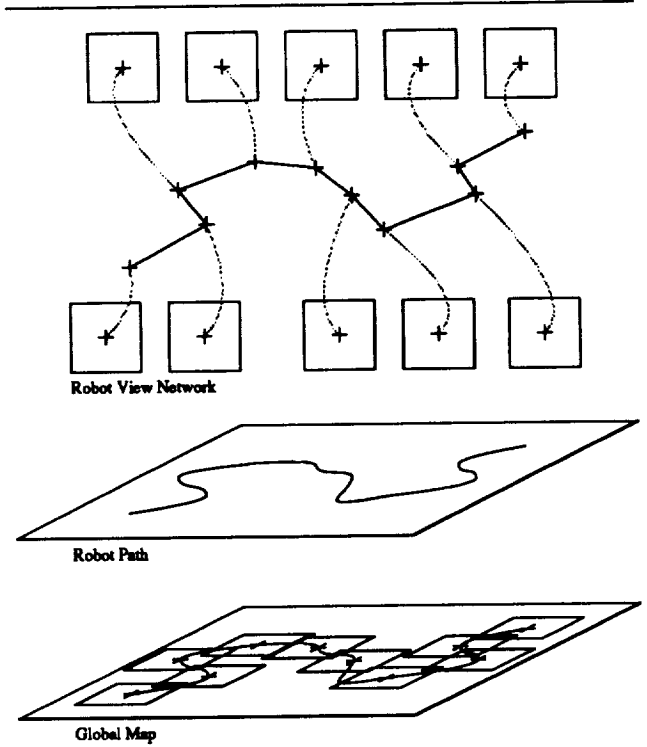


Figure 9: Maintaining a Dual Representation. A stochastic graph with the individual Robot Views is maintained in conjunction with the Global Map.

## 5 Other Applications

In the previous sections, we have seen that Occupancy Grids provide a unified approach to a number of issues in Robotics and Computer Vision. Additional tasks that can be addressed include the recovery of geometric descriptions from Occupancy Grids [7, 8], incorporation of user-provided maps, landmark recognition [8], prediction of sensor readings from Occupancy Grids, detection of moving objects using space-time filtering techniques, and other problems. In our own work, we are starting to explore two issues: the generation of 3D Occupancy Grids from depth profiles derived from laser scanners or stereo systems, and the development of mapping and navigation strategies that incorporate high-level user-provided maps when these are available.

It should be noticed that several robotic tasks can be performed on Occupancy Grids using operations that are similar or equivalent to computations performed in the image processing domain. Table 10 provides a qualitative overview and comparison of some of these operations.

We finalize our discussion with an observation concerning low-level versus high-level representations. It is interesting to observe that in Robotics and Computer Vision there has been historically a slow move from very high-level (stylized) representations of blocks-world objects to

Comparison of Operations on Occupancy Grids and on Images	
Occupancy Grids	Images
Labelling cells as Occupied, Empty or Unknown	Thresholding
Handling Position Uncertainty	Blurring/Convolution
Removing Spurious Spatial Readings	Low-Pass Filtering
Motion Solving/Map Matching	Correlation
Obstacle Growing for Path Planning	Region Growing
Path-Planning	Edge Tracking
Determining Object Boundaries	Edge Detection
Extracting and Labelling Occupied and Empty Areas	Segmentation/Region Colouring/Labeling
Prediction of Sensor Readings from User-Provided Maps	Convolution
Incorporating User-Provided Maps	Scan-Conversion
Object Motion Detection over Map Sequences	Space-Time Filtering

Figure 10: An Overview of Operations on Occupancy Grids and the Corresponding Image Processing Operations.

the recovery of simple spatial features in very constrained real images; from there to the recovery of surface patches; and recently towards "denser", tessellated representations of spatial information. A parallel evolution from sparse, high-level or exact descriptions to denser, lower-level and sometimes approximate descriptions can be seen in some other computational fields, such as Computer Graphics and Finite Element Analysis.

## 6 Conclusions

We have reviewed in this paper the Occupancy Grid framework and presented results from its application to mobile robot mapping and navigation in unknown and unstructured environments. The Occupancy Grid approach supports agile and robust sensor interpretation methods, incremental discovery procedures, composition of information from multiple sensors and over multiple positions of the robot, and explicit handling of uncertainty. Furthermore, the world models recovered using sensor data can be used efficiently in robotic planning and problem-solving activities. The results lead us to suggest that the Occupancy Grid framework provides an intermediate-level spatial representation that has the characteristics of robustness and generality necessary for real-world robotic applications.

## Acknowledgments

The author wishes to thank José Moura, Michael Meyer, Larry Matthies, Peter Cheeseman, Radu Jasinschi and Hans Moravec for their comments and suggestions concerning some of the issues discussed in this paper.

This research was supported in part by the Office of Naval Research under Contract N00014-81-K-0503. The author was supported in part by a graduate fellowship from the Conselho Nacional de Desenvolvimento Científico e Tecnológico - CNPq, Brazil, under Grant 200.986-80, and in part by the Mobile Robot Lab, CMU. The Field Robotics Center, CMU, provided travel support.

The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the funding agencies.

## References

- [1] J. O. Berger. *Statistical Decision Theory and Bayesian Analysis*. Springer-Verlag, Berlin, 1985. Second Edition.
- [2] A. E. Bryson and Y. C. Ho. *Applied Optimal Control*. Blaisdell Publishing Co., Waltham, MA, 1969.
- [3] A. Elfes. Handling Local and Global Maps Under Motion Uncertainty by a Mobile Robot. In *Proceedings of the 1988 IEEE International Symposium on Intelligent Control*, IEEE, Arlington, VA, August 1988.
- [4] A. Elfes. High-Level and Low-Level Spatial Models for Mobile Robot Mapping and Navigation. *IEEE Computer Magazine, Special Issue on Autonomous Intelligent Machines*, June 1989. To appear.
- [5] A. Elfes. Multiple Levels of Representation and Problem-Solving Using Maps From Sonar Data. In *Proceedings of the DOE/CESAR Workshop on Planning and Sensing for Autonomous Navigation*, Oak Ridge National Laboratory, UCLA, Los Angeles, August 18-19 1985.
- [6] A. Elfes. *The Occupancy Grid Approach to Robot Perception and Navigation*. PhD thesis, Electrical and Computer Engineering Department/Robotics Institute, Carnegie-Mellon University, 1989. To appear.
- [7] A. Elfes. A Sonar-Based Mapping and Navigation System. In *1986 IEEE International Conference on Robotics and Automation*, IEEE, San Francisco, CA, April 7-10 1986.
- [8] A. Elfes. Sonar-Based Real-World Mapping and Navigation. *IEEE Journal of Robotics and Automation*, RA-3(3), June 1987.
- [9] A. Elfes and L. Matthies. Sensor Integration for Robot Navigation: Combining Sonar and Stereo Range Data in a Grid-Based Representation. In *Proceedings of the 26th IEEE Conference on Decision and Control*, IEEE, Los Angeles, CA, December 9 - 11 1987.
- [10] S. Geman and D. Geman. Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(6), November 1984.
- [11] D. J. Kriegman, E. Triendl, and T. O. Binford. A Mobile Robot: Sensing, Planning and Locomotion. In *Proceedings of the 1987 IEEE International Conference on Robotics and Automation*, IEEE, Raleigh, NC, April 1987.
- [12] J. L. Marroquin. *Probabilistic Solution of Inverse Problems*. PhD thesis, Artificial Intelligence Lab, Computer Science Department, Massachusetts Institute of Technology, September 1985.
- [13] L. Matthies and A. Elfes. Integration of Sonar and Stereo Range Data Using a Grid-Based Representation. In *Proceedings of the 1988 IEEE International Conference on Robotics and Automation*, IEEE, Philadelphia, PA, April 25 - 29 1988.
- [14] H.P. Moravec and A. Elfes. High-Resolution Maps from Wide-Angle Sonar. In *IEEE International Conference on Robotics and Automation*, IEEE, St. Louis, March 1985. Also published in the 1985 ASME International Conference on Computers in Engineering, Boston, August 1985.
- [15] A. Papoulis. *Probability, Random Variables, and Stochastic Processes*. McGraw-Hill, New York, 1984.
- [16] R. C. Smith and P. Cheeseman. On the Representation and Estimation of Spatial Uncertainty. *The International Journal of Robotics Research*, 5(4), Winter 1986.
- [17] R. C. Smith, M. Self, and P. Cheeseman. A Stochastic Map for Uncertain Spatial Relationships. In *Proceedings of the 1987 International Symposium on Robotics Research*, MIT Press, 1987.
- [18] H. L. Van Trees. *Detection, Estimation, and Modulation Theory*. Volume Part I, John Wiley and Sons, New York, N.Y., 1968.
- [19] E. Vanmarcke. *Random Fields: Analysis and Synthesis*. MIT Press, Cambridge, MA, 1983.

## **NASA AMES RESEARCH CENTER**



# A Survey of Planning and Scheduling Research at the NASA Ames Research Center

Monte Zweben  
NASA Ames Research Center  
Moffett Field, CA 94035  
(415) 694-6940  
zweben@pluto.arc.nasa.gov

## Abstract

NASA Ames Research Center has a diverse program in planning and scheduling. This paper highlights some of our research projects as well as some of our applications. Topics addressed include machine learning techniques, action representations and constraint-based scheduling systems. The applications discussed are planetary rovers, Hubble Space Telescope scheduling, and Pioneer Venus orbit scheduling.

## 1 Introduction

NASA Ames Research Center's Artificial Intelligence Research Branch, led by Dr. Peter Friedland, has a diverse research program in planning and scheduling. Our work ranges from state-of-art fundamental research to applications of both new and existing technology. This paper is intended to summarize and highlight some of these activities.

The research issues we will highlight include: machine learning and planning, planning representations, non-symbolic representations, constraint-based scheduling, and the representation of procedural knowledge.

The applications we will present include Hubble Space Telescope scheduling, Mars Rover planning and scheduling, and Pioneer Venus orbit scheduling.

## 2 Planning and Scheduling

It is important to clarify the terms "planning" and "scheduling" before we proceed. An agent *plans* by finding actions that will take it from its current state to another desired state. Classically, this is a goal directed search through a space of possible partial plans. *Scheduling*, on the other hand, refers to an agent placing explicit times or orderings on a set of intended actions. This is usually a search through a space of possible timelines. In short, we call the process of finding actions that achieve goals *planning* and we call the placement of times on those actions *scheduling*.

## 3 Research

Our research program is a mix of internal research, university grants, and commercial contracts. Here we will present a representative subset of the program conducted at Ames, SRI, Stanford, and Carnegie-Mellon.

### 3.1 Learning in Planning

One of our group's areas of focus is machine learning and we are particularly interested in its application to planning and scheduling. We are exploring ways to improve search performance through the application of explanation-based learning techniques [Mit87, DeJ87]. The main idea behind this work is that a system can improve its performance by analyzing the solutions to problems it has previously encountered. As a result of this analysis, the system can remember the good decisions it made

as well as the poor ones. Ideally, we would like the system to generalize from this analysis so that the knowledge gained from its retrospection will be useful in cases that are not only identical to the ones it encountered, but also those that are close enough so that the previous experience would prove relevant and helpful.

Dr. Steven Minton, of Carnegie-Mellon University, performed a thorough analysis of a planning and learning system called PRODIGY [Min87,Min88]. PRODIGY is a STRIPS-like planner that employs explanation-based learning to acquire search control knowledge. His results showed that learning will not necessarily improve the performance of a planning system and in many cases it can degrade performance. As a result, Dr. Minton explored various methods of monitoring the utility of learned knowledge in order to transform (or possibly remove) learned knowledge to make the overall system more useful. Dr. Minton has recently joined our laboratory and will continue exploring planning and learning issues.

Another project within our laboratory is also addressing the utility problem in planning systems that learn. Monte Zweben and collaborators at the MITRE Corporation are specifically addressing the utility problem caused by the complexity of learned knowledge [Zwe88b]. When a planning system needs to make a decision it must consider the generalized information that it has learned. This pattern-matching overhead can overwhelm the system to the point where learned knowledge no longer aids efficiency. Using PRODIGY as a model, Mr. Zweben and his colleagues are developing a system that employs explanation-based learning (EBL) to acquire search knowledge, but relaxes some of the constraints usually associated with EBL techniques. Specifically, EBL generalizes from a single instance and guarantees the correctness of the learned knowledge. As a result, the learned information tends to be quite complex. This project's main extension to the PRODIGY model is the approximation of learned knowledge in the interest of lowering the expense of the relevancy check. As a result, this approximation of learned knowledge could be incorrect and must be monitored. If the learned knowledge is approximated erroneously and misleads the planner frequently, then the approximations must be refined. The goal of this project is to determine the approximation and refinement strategies that will result

in an efficient and effective collection of knowledge learned by an explanation-based component.

### 3.2 Planning Representations

Dr. Mark Drummond, of our group, takes a Net Theory approach to the problem of planning, scheduling and control [Dru85,Dru87]. His approach has a number of interesting features and advantages. Similar to Amy Lansky's [Lan87] work, it views a plan as a set of constraints over a pre-specified set of actions. Unlike Lansky's GEM model, however, the Net Theory approach allows one to distinguish clearly between orderings required by causality, and those that are simply convenient, given the agent's goals. The Net Theory approach also begins to make clear the true role of *least commitment planning*, where orderings on actions are postponed until an ordering decision *must* be made. Current plan representations frequently over-commit to specific orderings. This over-commitment is critical when dealing with complicated scheduling problems, since many orderings and conditions cannot be determined until a schedule is actually being carried out. The Net Theory approach currently being explored by Dr. Drummond allows complete postponement of ordering decisions until all environmentally determined information is available. This permits a new view on the role of an agent's synthetic temporal data structure. These data structures can now be viewed as plans, schedules, or control programs, depending on the phase of overall system operation. This work does not view planning and scheduling as a one-time process, but rather, includes an explicit control phase where plans/schedules are incrementally modified to suit execution needs.

Dr. Drummond is also exploring a number of other issues in his planning research including: the tradeoff of reactive and predictive scheduling, the role of means-ends analysis in planning, the integration of planning and scheduling mechanisms, the representation and derivation of conditional and iterative plans, the role of constraint-satisfaction in the planning process, and the use of domain constraints to control planning search [Dru88].



### 3.3 Control Without Symbols

The work of Dr. Stan Rosenchein, formerly of SRI International and now of Teleos Research, takes the perspective that expensive symbolic processing at run time can be avoided by compiling symbolic representations into circuitry guaranteed to act in bounded time. Dr. Rosenchein and his colleague Leslie Kaelbling have developed a set of tools that enables one to design a robotic controller in a high-level language, which then gets compiled into efficient circuitry that can be simulated or manufactured in hardware [Kae88,Ros86]. The fundamental idea behind this work is that much of the expensive search (like pattern matching) employed by symbolic reasoners can be accomplished at compile time, allowing the robot to quickly process its sensory information and react appropriately. One of their tools, Gapps [Kae88], takes a goal expression and rules in a goal decomposition language and outputs circuitry that will enable a system to take action given a goal and its current state. Their tool REX allows one to specify behavior that takes sensory input and the system's current state and updates the current state to reflect what has occurred in the system's environment. REX allows one to specify the circuitry in a language more abstract than circuits, but less abstract than that of a programming language. They are currently designing a system called RULER which will allow one to design the state update circuitry in a logical language resembling PROLOG. Ultimately, this language will be compiled into REX specifications.

This work is distinguished in that the REX language has been specifically designed to support analysis of any particular REX program to prove its correctness. Further, this work is currently used to control Flakey, the SRI mobile robot. We view this work as a realistic first step towards the production of efficient robotic control tools. It begins to show how a designer can allocate computational resources at different phases of the design and deployment process.

### 3.4 Constraint-based Scheduling

As previously mentioned, scheduling is the process of placing a pre-specified set of actions on a timeline ensuring that the schedule's constraints are maintained. One of our projects, led by Monte Zweben, addresses the formulation and resolu-

tion of complex scheduling and resource allocation problems using constraints to represent scheduling knowledge and preferences [Zwe88a]. Constraints are declarative representations of relationships that abstract away control flow. They allow one to specify the relationships between the problem's variables in a system and enable the system to automatically determine the computation path from known variables to the unknown [Sta77]. These representations can be used for lookahead in a search process. Lookahead or constraint propagation results in less backtracking (i.e., fewer futile search paths) because commitments to various choices in the system are made only if they are compatible with the choices remaining in the system [Har80,Ste80]. However, lookahead can result in unnecessary constraint propagation. To circumvent this problem, we employ a technique called delayed evaluation [Fil84]. A system employing delayed evaluation does not completely evaluate its data structures until they are accessed. We use the data structure *streams* [Abe85] which are lists that delay the evaluation of their tails (i.e., all the elements of the list except the first element). The use of streams is advantageous for two main reasons: 1) their delayed evaluation circumvents unnecessary constraint propagation; 2) their delayed evaluation is transparent to knowledge engineers because stream operations are quite similar to list operations and our model of constraint-satisfaction is based upon list operations.

### 3.5 Procedural Knowledge

Dr. Michael Georgeff of SRI International has developed a system called PRS - Procedural Reasoning System - that enables one to represent and use complex procedural knowledge [Geo86]. PRS takes a set of procedures and executes them in a goal-directed manner. It uses a declarative representation of procedures that extends the expressiveness of previous action representations. Actions in PRS can exhibit iteration and recursion and also can employ run-time conditional branching. Thus, decisions as to what action to perform next can be dependent upon the runtime environment. PRS procedures can also be interrupted by other procedures, thereby allowing emergency recognition and exception handling. The ability to change its focus of attention quickly and to act conditionally makes PRS a highly reactive system.

PRS also has interesting theoretical aspects in that it meets much of the rational agency criteria proposed in the recent philosophical literature. Because PRS behaves like a rational agent there is potential for the development of interesting explanation components. PRS has been exercised in a very complex and interesting domain: malfunction handling for the reaction control system of the Space Shuttle. NASA diagnostic manuals were encoded in PRS resulting in an extensible set of semi-autonomous procedures.

## 4 Applications

The Ames AI Research Lab performs state-of-the-art research, but does so in the context of real-world applications. This allows us to both verify that our methods scale-up to real problems and focus our research towards topics of interest to NASA. In addition to framing our research within NASA problems, we also demonstrate the utility of known AI techniques with engineering applications. Don Rosenthal is the director of our applications work. His applications projects include Pioneer Venus satellite scheduling and Hubble Space Telescope scheduling. In fiscal year 1989, Mr. Rosenthal will explore planetary rover applications.

### 4.1 Pioneer Venus

This project, now completed, showed the utility of rule-based systems for operational software [Ros88]. We developed a heuristic ground-based scheduler for science operations (e.g., instrument configurations, data storage and playback, telemetry, etc.) onboard the Pioneer Venus satellite. This software is currently performing a task in minutes which formerly took people hours. Further, the resulting schedules are as effective as the man-made ones but contain fewer flaws. The satellite's operations are currently scheduled with this expert system. This scheduler is the first expert system installed in day to day use within a NASA mission operations environment.

### 4.2 Hubble Space Telescope (HST) Scheduling

Thousands of proposed observations for HST must be processed by the Space Telescope Science In-

stitute (STScI), on the Johns Hopkins University campus in Baltimore, to construct schedules for the science operations of the orbiting optical observatory. Current software is not flexible or extensible enough to meet the operational demands expected on the system and we are helping to provide knowledge-based solutions to this problem [Mil87].

The HST projects we support take a constraint-based approach to scheduling. Dr. Stephen Smith, of Carnegie-Mellon University, is applying research in factory scheduling [Fox83,Smi86] to the HST problem. This approach is well suited for over-constrained problems where a solution requires the relaxation of constraints.

Another project, at the STScI, is applying state-of-the-art constraint satisfaction techniques to the HST scheduling problem. Their goal is to produce a flexible and extensible scheduler that can dynamically react to anomalies and re-schedule accordingly. This work has resulted in a program called SPIKE, which uses piecewise constant functions to quantitatively represent the degree of constraint violation. Using these functions, SPIKE can efficiently combine constraints as well as judge the options it must choose.

### 4.3 Planetary Rovers

In the coming year we will begin performing extensive research into the planetary rover problem while concentrating on the science planning and scheduling issues. Using the Mars Rover domain as a model, we are interested in rovers that can autonomously plan and execute an appropriate set of scientific analyses for many different science goals. Further, we will explore techniques that dynamically discover interesting science opportunities, and attempt to replan the rover's actions to accommodate these new goals.

Additionally, we will address the integration of navigation planning and science planning which will require research in systems that negotiate for resources and time.

We will also explore machine learning techniques that can improve the overall rover system. First, we will explore techniques that improve a system's search performance. Second, we will address model refinement for rovers that begin with a rough and incomplete model of their environment. These techniques review a system's actions and remember when they succeed and when they fail. They

also find discrepancies between a system's expectations and its observations and uses these discrepancies to refine the system's models.

## 5 Summary

This paper is intended to selectively introduce our research and to point out references to technical papers. Some of the areas currently addressed by our group but not discussed here are: 1) planning with incomplete models [Car87b,Car87a], 2) the use of truth-maintenance in planning [Mor86], and 3) communicating, cooperating agents [Nil87]. In the coming year, we plan to expand our efforts in multi-agent planning and constraint satisfaction. The overall goal of the program is to develop the technology for large-scale automation of space missions.

## 6 Acknowledgements

Thanks to Mark Drummond, Peter Friedland, Steve Minton, Don Rosenthal, and Haym Hirsh for their contributions to this survey.

## References

- [Abe85] Abelson, H., and Sussman, G.J. *Structure and Interpretation of Computer Programs*. MIT Press, Cambridge, MA, 1985.
- [Car87a] Carbonell, J.C. and Gil, Y. Learning by Experimentation. In *Proceedings of the Fourth International Workshop on Machine Learning*, Morgan Kaufman, Palo Alto, CA, 1987.
- [Car87b] Carbonell, J.G., Mason, Matthew T., and Mitchell, Tom M. Machine Learning and Planning in Reactive Environments. In *NASA Ames AI Forum Proceedings*, 1987.
- [DeJ87] DeJong, G.F. and Mooney, R. Explanation-Based Generalization: An Alternative View. *Machine Learning*, 1(2), 1987.
- [Dru85] Drummond, M.E. Refining and extending the procedural net. In *IJCAI-9 Proceedings*, Morgan Kaufman, Palo Alto, CA, 1985.
- [Dru87] Drummond, Mark E. A Representation of Action and Belief for Automatic Planning Systems. In *Proceedings of the AAAI/CSLI 1986 Workshop on Reasoning about Actions and Plans*, Morgan Kaufman, Palo Alto, CA, 1987.
- [Dru88] Drummond, M.E., and Currie, K.W. Exploiting temporal coherence in non-linear plan construction. *Computational Intelligence Journal*, to appear, 1988.
- [Fil84] Filman, R.E., Friedman, D.P. *Coordinated Computing: Tools and Techniques for Distributed Software*. McGraw-Hill Book Company, New York, NY, 1984.
- [Fox83] Fox, Mark S. *Constraint-Directed Search: A Case Study of Job Shop Scheduling*. PhD thesis, Carnegie-Mellon University, 1983.
- [Geo86] Georgeff, M.P. and Lansky, A.L. Procedural Knowledge. *Proceedings of the IEEE*, Special Issue on Knowledge Representation, 1986.
- [Har80] Haralick, R.M., Elliot, G.L. Increasing Tree Efficiency for Constraint Satisfaction Problems. *Artificial Intelligence*, 14, 1980.
- [Kae88] Kaelbling, L.P. Goals As Parallel Program Specifications. In *AAAI-88*, Morgan Kaufman, Palo Alto, CA, 1988.
- [Lan87] Lansky, Amy L. A Representation of Parallel Activity Based on Events, Structure, and Causality. In *Proceedings of the 1986 Workshop on Reasoning about Actions and Plans*, Morgan Kaufman, Palo Alto, CA, 1987.
- [Mil87] Miller, G., Rosenthal, D., Cohen, W., and Johnston, M. Expert Systems Tools for Hubble Space Telescope Observation Scheduling. *Telematics and Informatics*, 4(4), 1987.

- [Min87] Minton, S. and Carbonell, J.G. Strategies for Learning Search Control Rules: An Explanation-based Approach. In *IJCAI-87 Proceedings*, Morgan Kaufman, Palo Alto, CA, 1987.
- [Min88] Minton, S. Empirical Results Concerning the Utility of Explanation-based Learning. In *AAAI-88 Proceedings*, 1988.
- [Mit87] Mitchell, T.M., Keller, R.M., and Kedar-Cabelli, S.T. Explanation-Based Generalisation: A unifying view. *Machine Learning*, 1(1), 1987.
- [Mor86] Morris, P.H. and Nado, B. Representing Actions with an Assumption-based Truth-Maintenance System. In *AAAI-86*, 1986.
- [Nil87] Nilsson, Nils J., Cohen, Phillip R., Rosenchein, Stanley J., and Fertig, Kenneth. Intelligent Communicating Agents. In *NASA Ames AI Forum Proceedings*, 1987.
- [Ros86] Rosenchein, S.J. and Kaelbling, L.P. The Synthesis of Digital Machines with Provable Epistemic Properties. In *Proceedings of the Conference on Theoretical Aspects of Reasoning and Knowledge*, Morgan Kaufman, Palo Alto, CA, 1986.
- [Ros88] Rosenthal, D.A. and Jackson, Robert W. Development of a Pioneer Venus Expert Scheduling System. In *Proceedings of the 26th Aerospace Sciences Meeting AIAA-88*, 1988.
- [Smi86] Smith, S.F., Fox, M.S., and Ow, P.S. Constructing and Maintaining Detailed Production Plans: Investigations into the development of Knowledge-based Factory Scheduling Systems. *AI Magazine*, 7(4), 1986.
- [Sta77] Stallman, R.M., Sussman, G.J. Forward Reasoning and Dependency-Directed Backtracking in a System for Computer-Aided Circuit Analysis. *Artificial Intelligence*, 9, 1977.
- [Ste80] Stefik, Mark. *Planning With Constraints*. PhD thesis, Stanford University, January 1980.
- [Zwe88a] Zweben, M., Eskey, M., and Rosenthal, D. Constraint Satisfaction with Delayed Evaluation. *Submitted to CACM*, 1988.
- [Zwe88b] Zweben, Monte and Chase, Melissa P. Improving Operationality with Approximate Heuristics. In *Proceedings of the 1988 Spring Symposia Series in Explanation-based Learning*, 1988.

# Integrating Planning and Reactive Control\*

Stanley J. Rosenschein  
Leslie Pack Kaelbling  
Teleos Research  
576 Middlefield Road  
Palo Alto, CA 94301

## 1 Introduction

Artificial intelligence research on planning is concerned with designing control systems that choose actions by manipulating explicit descriptions of the world state, the goal to be achieved, and the effects of elementary operations available to the system. Because planning shifts much of the burden of reasoning to the machine, it holds great appeal as a high-level programming method [3,10,12]. Experience shows, however, that it cannot be used indiscriminately because even moderately rich languages for describing goals, states, and the elementary operators lead to computational inefficiencies that render the approach unsuitable for realistic applications. This inadequacy has spawned a recent wave of research on "reactive control" or "situated activity" in which control systems are modeled as reacting directly to the current situation rather than as reasoning about the future effects of alternative action sequences [2,1,11]. While this research has confronted the issue of run-time tractability head on, in many cases it has done so by sacrificing the advantages of declarative planning techniques.

This paper discusses ways in which the two approaches can be unified. We begin by modeling reactive control systems as state machines that map a stream of sensory inputs to a stream of control outputs. These machines can be decomposed into two continuously active subsystems: the planner and the execution module. The planner computes a "plan," which can be seen as a set of bits that control the behavior of the execution module. An important element of this work is the formulation of a precise semantic interpretation for the inputs and outputs of the planning system. We show that the distinction between planned and reactive behavior is largely in the eye of the beholder: Systems that seem to compute explicit plans can be redescribed in situation-action terms and vice versa. We also discuss practical programming techniques that allow the advantages of declarative programming and guaranteed reactive response to be achieved simultaneously.

---

\*This work was supported in part by NASA Cooperative Agreement #NCC-2-494 through Stanford subcontract #PR6359 and in part by a gift from the System Development Foundation.

## 2 Planning and Reactive Control

Classical AI views the generation of behavior as a two-step process consisting of planning and execution. Planning produces a data structure describing a course of action; execution is the step-by-step interpretation of this data structure to produce overt behavior. The planning step can be viewed as a form of stylized program synthesis in a weak logic of programs, and many formalisms have been proposed to capture the logic of planning. A common approach is to employ predicate calculus formulas as state descriptions (e.g.,  $on(blockA, blockB)$ ) and to model operators as state-transforming functions, described either axiomatically (using facts of the form  $holds(p, s) \rightarrow holds(q, op(s))$ ) or as syntactic transformations that map state descriptions to state descriptions. Letting  $ops$ ,  $init$ , and  $goal$  stand for formulas expressing, respectively, facts about the operators, the initial conditions, and the goal statement, we require the planner to find  $plan = make\_plan(ops, init, goal)$  such that

$$ops \models init \wedge plan \rightarrow goal .$$

In other words, it should follow from the operator descriptions that if the initial condition holds and the plan is carried out, the goal condition will be achieved. Note also that  $init \wedge plan$  should be consistent; otherwise, the requirement can be trivially satisfied.

The complexity of plan synthesis obviously depends on the specific nature of the domain. For realistic domains, however, traditional planning typically requires significantly more time than the fundamental reflex cycle of the system, and controlling the rate at which planning occurs relative to changes in the environment is extremely challenging. For this reason, classical planning techniques have almost always been applied, in practice, to “static” domains, in which the only significant source of change is the agent itself and in which, therefore, the time required for planning can be safely ignored.

In an attempt to deal with more dynamic domains, some researchers have abandoned planning in favor of reactive control, which does not take a two-stage view of behavior generation. In this approach, the behavior of the agent is specified directly using situation-action rules that are evaluated at frequent intervals. A reactive control system could be implemented, for example, as a program executing a tight loop, the body of which exhibits a high degree of conditionality, for example:

```
do forever
  if tiger_approaching then
    set wheel velocities to [+30,+30],
  else if ...
```

Since the conditions can be evaluated in parallel, reactive systems can also be described as circuits or operator networks implementing a function that maps a stream of information states to a stream of output commands to the effectors. The key to reactivity is to design this function so that it can be computed quickly again and again.

Each approach has its advantages. Planning provides a convenient high-level declarative formalism and leaves much of the reasoning to the machine. In principle, this makes it possible for the control system to handle classes of situations that are too complex for the programmer to anticipate in advance but are amenable to analysis at run time, once a concrete initial state and goal state are available. In contrast, reactive control offers the advantage of guaranteed

response time and hence the ability to react quickly to a changing environment. Because neither approach clearly dominates the other and because many application domains have attributes that make each attractive, a synthesis of these two techniques is necessary.

One method for achieving such a synthesis is to embed a reactive controller in a classical planner-based architecture. In a sense, this is what the term “execution monitoring” is often taken to mean in classical planning: The planner sends a data structure to the execution module, which in turn reacts to changing world conditions under the control of the plan. The execution module is also able to detect conditions in the world that violate the assumptions upon which the plan’s correctness depends. Unfortunately, the mathematical framework of classical planning, based on atemporal state transformations, offers little guidance as to how the passage of time during the planning process ought to be handled.

Since reactive control is based on a model of time-bounded computation, it is more natural to incorporate planning by extending the reactive-control architecture rather than vice versa, and this is the approach we shall take. In order to do this, however, we must first characterize the semantics of the data structures produced by the planner in a way that makes sense in the reactive control model.

### 3 Semantics for Planning and Control

We shall model a control system as a state machine that transduces inputs carrying information about the environment to outputs that affect the environment. In the simplest case, this machine has no state and simply computes a pure function from inputs to outputs. In more complex cases, including cases in which significant planning occurs, the computation requires internal state. A major challenge in designing control systems is to provide a clear semantic model of the information available to the control system, of the goals achieved by the chosen actions, and of the mapping between the two.

Let  $M$  be a control system with input variable  $in$ , output variable  $out$ , and an internal state vector  $a$ . The inputs carry information about the world, the outputs are commands to the effectors, and the internal state allows the computation of outputs to depend on past inputs and to be extended in time. To introduce a planner into this model, we decompose the machine into components, introducing three subsidiary variables,  $init$ ,  $goal$ , and  $plan$ , and four sub-machines:  $E_{init}$ ,  $E_{goal}$ ,  $Planner$ , and  $Exec$ . We assume that  $ops$  is fixed in advance. The inputs and outputs of these modules are as follows:

- $E_{init}$ : input  $in$ , output  $init$
- $E_{goal}$ : input  $in$ , output  $goal$
- $Planner$ : input  $init$ ,  $goal$ , output  $plan$
- $Exec$ : input  $in$ ,  $plan$ , output  $out$

The overall structure of the machine is illustrated in Figure 1. Informally, the  $E_{init}$  and  $E_{goal}$  machines operate on the input, extracting values representing the initial conditions and goal condition, respectively. These are transduced by  $Planner$ , in a way that may involve internal state and computation over time, to a continuously available  $plan$  output. Note, however,

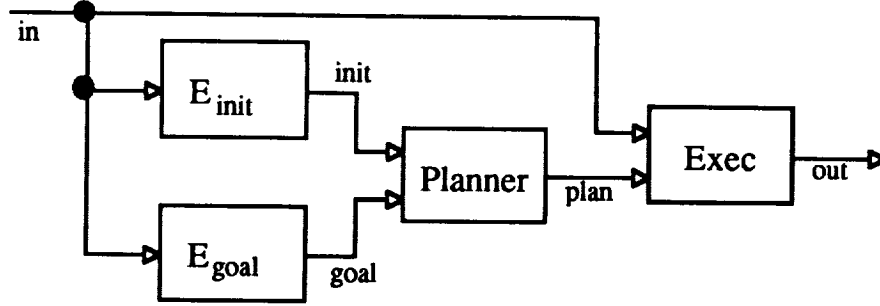


Figure 1: Embedding a planner in a reactive control system.

that the output may be vacuous, indicating that the final plan has not yet been computed [4].

We are interested in characterizing the semantics of the inputs of *Planner* and of its result, but must first consider the more general question of where semantic “interpretations” for data values come from.

For data structures like *init*, the classical view is that the data value is a description of facts about the world expressed in some language whose semantics is clear to the designer of the system. This description would be of little use were it not also the case that when the data structure had a particular value, the condition denoted was guaranteed to hold in the environment. Such semantic considerations form the foundation of the situated-automata model in which the semantics of data structures are characterized in terms of objective correlations with external reality rather than in terms of designer-stipulated interpretations. In this approach, one says a machine variable  $x$  carries the information that  $p$  in world state  $s$ , written  $s \models K(x, p)$ , if for all world states in which  $x$  has the same value it does in  $s$ , the proposition  $p$  is true. The formal properties of this model and its usefulness for programming embedded systems have been described elsewhere [7,8,5,9].

Since we are committed to an information-based semantics for reactive systems, we seek an “objective” semantics of goals defined explicitly in informational terms. We can reformulate the notion of having a goal  $p$  as having the information that  $p$  implies a fixed top-level goal, called  $N$  for “Nirvana.” Formally, we define a goal operator  $G$  as follows:

$$G(x, p) \equiv K(x, p \rightarrow N)$$

In this model,  $x$  has the goal  $p$  if  $x$  carries the information that  $p$  implies Nirvana.<sup>1</sup> Since this defines goals explicitly in terms of information, the same formal tools used to study information can be applied to goals as well. In fact, under this definition, goals and information are dual concepts.

To see this, consider a function  $f$  mapping values of one variable,  $a$ , to values of another variable,  $b$ . Under the information interpretation, such a function takes elements having more specific information into elements having less specific information. This is because functions generally introduce ambiguity by mapping distinct inputs to the same output. For example, if value  $u_1$  at  $a$  is correlated with proposition  $p$  and value  $u_2$  at  $a$  is correlated with  $q$  and if

<sup>1</sup>We observe that under this definition *False* will always be a goal; in practice, however, we are only interested in non-trivial goals.



$f$  maps both  $u_1$  and  $u_2$  to  $v$  at  $b$ , the value  $v$  is ambiguous as to whether it arose from  $u_1$  or  $u_2$ , and hence the information it contains is the disjunctive information  $p \vee q$ , which is less specific than the information contained in either  $u_1$  or  $u_2$ . Thus, functional mappings are a form of forgetting.

Under the goal interpretation, this picture is reversed. The analog to “forgetting” is committing to subgoals, which can be thought of as “forgetting” that there are other ways of achieving the condition. For instance, let the objective information at variable  $a$  be that the agent is hungry and that there is a sandwich in the right drawer and an apple in the left. If the application of a many-to-one function results in variable  $b$ ’s having a value compatible with the agent’s being hungry and there being a sandwich in the right drawer and either an apple in the left drawer or not, we could describe this state of affairs by saying that variable  $b$  has lost the information that opening the left drawer would be a way of finding food. Alternatively, we could say that variable  $b$  had committed to the subgoal of opening the right drawer. The phenomena of forgetting and commitment are two sides of the same coin.

Formally we can relate this observation to axioms describing information and goals. One of the formal properties satisfied by  $K$  is the deductive closure axiom, which can be written as follows:

$$K(x, p \rightarrow q) \rightarrow (K(x, p) \rightarrow K(x, q)) .$$

The analogous axiom for goals is

$$K(x, p \rightarrow q) \rightarrow (G(x, q) \rightarrow G(x, p)) .$$

This is precisely the subgoaling axiom. If the agent has  $q$  as a goal and carries the information that  $q$  is implied by some other, more specific, condition,  $p$ , the agent is justified in adopting  $p$  as a goal. The validity of this axiom can be established directly from the definition of  $G$ .

Given these two ways of viewing the semantics of data structures, we can revisit the *Planner* module with inputs *init* and *goal* and output *plan*. The most natural way to interpret the values of these variables is to apply the information interpretation to the values of *init* and the goal interpretation to the values of *goal* and *plan*. However, as observed above, since the goal interpretation is derived directly from the informational model, we could have applied either interpretation to any of the values.

In summary, one need not think of “planning” as an essentially different kind of function performed by the system. Rather, it can be thought of as a *perspective* one takes on certain data structures when one thinks of them—for design convenience—as encoding goals rather than information.

## 4 Current Research Directions

In this section we list several efforts currently underway that are aimed at exploring the practical consequences of our approach toward integrating planning and reactive control.

### 4.1 Embedding Planning in Gapps

Gapps [6] is a declarative language for programming reactive systems. The Gapps compiler takes as input a top-level goal and a set of goal reduction rules and produces as output a

program for achieving the top-level goal. The program is guaranteed by construction to map information states to actions in constant time. By using Gapps, the programmer can gain many of the benefits of declarative programming without sacrificing real-time response. One direction of research is to embed planning in Gapps by converting operator descriptions into goal reduction rules, which in turn are transformed by Gapps into real-time programs. A typical rule schema might be:

```
(defgoalr (ach P)
  (if (regress P a)
      (do a)
      (ach (regress P a))))
```

Because Gapps produces a fixed-size circuit at compile time, a compile-time bound must be placed on the *depth* of the regression, although in principle the actual calculation of the regressed condition can be deferred to run time.

## 4.2 Temporally Extended Planning Processes

Traditional planners operate by carrying out a guided search through a space of plans. Depending on the combinatorics of the search, this process may or may not succeed within a single cycle of the reactive system. If it does not, the search must proceed in parallel with the execution of a more reactive, though perhaps less effective, behavior. Since the passage of time affects whether or not a data value will continue to be correlated with the environment, it is clear that the semantics of temporally-extended planning will be time-dependent. A simple solution to this problem is for the planner to monitor world conditions that would invalidate the current plan and to output the vacuous plan when those conditions arise [4]. While correct, this approach is not maximally information-preserving and more subtle methods are possible. In the case of informational data structures, we have explored declarative programming techniques to control the updating of the machine's information state so that maximal correlation with the environment is maintained [9], and similar methods might be applied to planning over time as well.

## 4.3 Trading Flexibility for Performance

As in conventional programming, some information required for action selection might be available at compile time, while other information may become available only at run time. Ease of programming would be enhanced by minimizing syntactic and semantic distinctions based only on differences as to when information becomes available. In traditional compilers, for instance, constant-folding optimizations take advantage of compile-time information about the values of expressions in a way that is entirely transparent to the programmer. For planning and control applications, this transparency is more difficult to achieve because without sufficient compile-time information, the symbolic synthesis procedure may not terminate, and without a clear compile-time versus run-time model in mind, the programmer may lack sufficient insight to adequately control the compilation process. Nevertheless, our ultimate goal is to make it as easy as possible to trade off flexibility against performance by conveniently moving the boundary between compile-time and run-time processing.

## Acknowledgments

We have benefited greatly from discussions with Mark Drummond and Monte Zweben.

## References

- [1] Agre, Philip E. and David Chapman. "Pengi: An Implementation of a Theory of Activity." *Proceedings of the Sixth National Conference on Artificial Intelligence*, Seattle, Washington (July 1987).
- [2] Brooks, Rodney A. "A Robust Layered Control System for a Mobile Robot." Technical Report 864, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts (1985).
- [3] Fikes, Richard and Nils J. Nilsson. "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving." *Artificial Intelligence*, Vol. 2, Nos. 3,4 (1971).
- [4] Kaelbling, Leslie P. "An Architecture for Intelligent Reactive Systems." In Michael P. Georgeff and Amy L. Lansky, editors, *Reasoning about Actions and Plans: Proceedings of the 1986 Workshop*. Morgan Kaufmann, Los Altos, California (1987).
- [5] Kaelbling, Leslie P. "Rex: A Symbolic Language for the Design and Parallel Implementation of Embedded Systems." *Proceedings of the AIAA Conference on Computers in Aerospace*, Wakefield, Massachusetts (1987).
- [6] Kaelbling, Leslie P. "Goals as Parallel Program Specifications." *Proceedings of the Seventh National Conference on Artificial Intelligence*, Morgan Kaufmann, St. Paul, Minnesota (August 1988).
- [7] Rosenschein, Stanley J. "Formal Theories of Knowledge in AI and Robotics". In *New Generation Computing*, Vol. 3, No. 4, (special issue on Knowledge Representation), Ohmsha, Ltd., Tokyo, Japan (1985).
- [8] Rosenschein, Stanley J. and Leslie P. Kaelbling. "The Synthesis of Digital Machines with Provable Epistemic Properties," *Proceedings of Workshop on Theoretical Aspects of Reasoning About Knowledge*, Monterey, California (1986).
- [9] Rosenschein, Stanley J. "Synthesizing Information-Tracking Automata from Environment Descriptions." *Proceedings of the First Conference on Principles of Knowledge Representation and Reasoning*, Toronto, Canada (to appear).
- [10] Sacerdoti, Earl. *A Structure for Plans and Behavior*. Elsevier North-Holland, Inc., New York (1977).
- [11] Schoppers, Marcel J. "Universal Plans for Reactive Robots in Unpredictable Environments." *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, Morgan Kauffman, Milan (1987).

- [12] Wilkins, David E. *Practical Planning: Extending the Classical AI Planning Paradigm*. Morgan Kaufmann Publishers, Inc. San Mateo, California (1988).

# LEARNING IN STOCHASTIC NEURAL NETWORKS FOR CONSTRAINT SATISFACTION PROBLEMS

**Mark D. Johnston**

*Space Telescope Science Institute<sup>1</sup>  
3700 San Martin Drive, Baltimore, MD 21218 USA*

**Hans-Martin Adorf**

*Space Telescope - European Coordinating Facility  
European Southern Observatory  
Karl-Schwarzschild-Str. 2, D-8046, Garching bei München, F.R. Germany*

## Abstract

We describe a newly-developed "artificial neural network" algorithm for solving constraint satisfaction problems (CSPs) which includes a learning component that can significantly improve the performance of the network from run to run. The network, referred to as the Guarded Discrete Stochastic (GDS) network, is based on the discrete Hopfield network but differs from it primarily in that auxiliary networks (guards) are asymmetrically coupled to the main network to enforce certain types of constraints. Although the presence of asymmetric connections implies that the network may not converge, we find that, for certain classes of problems, the network often quickly converges to find satisficing solutions when they exist. The network can run efficiently on serial machines and can find solutions to very large problems (e.g.  $N$ -queens for  $N$  as large as 1024). One advantage of the network architecture is that network connection strengths need not be instantiated when the network is established: they are needed only when a participating neural element transitions from off to on. We have exploited this feature to devise a learning algorithm, based on consistency techniques for discrete CSPs, that updates the network biases and connection strengths and thus improves the network performance.

## 1 Introduction

Constraint satisfaction problems (CSPs) arise frequently in AI applications and have been investigated by many researchers. Most of the commonly used methods for finding solutions to CSPs are based on backtracking tree search or its variants. A variety of techniques have been utilized to make this type of search more efficient: these include pre-processing the constraints, ordering the instantiation of variables, or making intelligent decisions about how to backtrack when a deadend is encountered (see, e.g., [1,2,3,4]).

A very different approach has been taken by researchers investigating "artificial neural network" or "connectionist" approaches to solving CSPs (e.g. [5,6]). In this type of approach the constraints are encoded in the

---

<sup>1</sup> Operated by the Association of Universities for Research in Astronomy for the National Aeronautics and Space Administration

network topology and connection strengths so that the state of the network when it converges can be interpreted as a solution to the CSP. The network dynamics can be described in terms of an “energy” function which the network minimizes as it runs [7]. A problem with these methods is the tendency of the network to settle into a local minimum of the energy function, representing a solution only to a sub-problem of the CSP. Techniques for escaping from local minima are known [8,9] but tend to be time-consuming and thus greatly limit the size of the problem that can be represented and solved. We have previously described a new network architecture which circumvents some of these problems [10]. Our approach, which we call the Guarded Discrete Stochastic (GDS) network, avoids local minima by coupling the main network to one or more fast-acting auxiliary (guard) networks that enforce additional higher-order constraints. While this has the drawback that the network is no longer guaranteed to converge to any stable configuration, we find that for a variety of problems the network has a high probability of converging with sufficient speed that solutions to very large problems can be found even on serial machines.

In the GDS network, as in other neural network approaches to CSPs, the problem is explicitly encoded in the network when it is constructed. This is in contrast to the use of neural networks on other types of problems where the network goes through a training phase to “learn” the values of the connection strengths and biases that are appropriate to the problem [9,11]. One advantage of the GDS network architecture and update scheme is that the connections can be treated as “virtual”, i.e. the values of the connection strengths are not needed until a participating neuron transitions from off to on. We have found that this can be used as the basis for a learning algorithm that infers additional constraints only from instantiated connections. This can be viewed as the network analog of “learning while searching” as successfully applied to backtracking tree search [12].

In the following (Section 2) we first briefly describe the GDS network architecture and update scheme from [10]. We then describe the learning algorithm and present results for two CSPs that show how learning can significantly improve the network’s performance (Section 3). We conclude with a general discussion of the network’s behavior during search and why the learning algorithm is effective (Section 4).

## 2 The Guarded Discrete Stochastic (GDS) Network

The problem we consider is a general binary CSP involving a set of  $N$  variables  $X_1, \dots, X_N$  with domains  $D_1, \dots, D_N$ , and an associated set a set of constraints  $C_\alpha(X_j, X_k), \alpha = 1, \dots, M$ . A binary constraint is a subset of the Cartesian product  $D_j \times D_k$  which specifies combinations of values which are incompatible with each other. A solution is an assignment of values to all of the variables so that no constraints are violated. We are interested here in the problem of finding at least one satisfying assignment (the satisficing problem).

We first consider how to represent this CSP by a Hopfield discrete neural network [7] of which the GDS network is a generalization. Let the output (zero or one) of the neuron labeled  $ij$  be denoted by  $y_{ij}$ , where  $i$  refers to the  $i^{\text{th}}$  variable  $X_i$  and  $j$  refers to  $d_{i,j}$ , the  $j^{\text{th}}$  value in the domain  $D_i$ . When viewed as a matrix (with a variable column width depending on cardinality of the domains  $D_i$ ), rows are associated with variables and columns are associated with values.

The assignment of  $d_{i,j}$  to  $X_i$  is represented by  $y_{ij} = 1$ . The input  $x_{ij}$  to neuron  $ij$  is the sum of a bias term  $b_{ij}$  and a weighted sum of the output of other neurons:

$$x_{ij} = \sum_{mn} W_{ij,mn} y_{mn} + b_{ij} \quad (1)$$

$W_{ij,mn}$  is called the connection matrix. In the two-state neuron model the output is related to the input by:

$$y_{ij} = \begin{cases} 1 & x_{ij} \geq 0 \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

In the discrete Hopfield model with no transmission delays, neurons are selected at random and their output is set according to Eqn. (2). When the connections are symmetric ( $W_{ij,mn} = W_{mn,ij}$ ) and there is no self-feedback ( $W_{ij,ij} = 0$ ), then there exists a bounded “energy” function which the network minimizes as it runs. The biases

$b_{ij}$  and connection weights  $W_{ij,mn}$  can be chosen so that a solution to the CSP is a minimum of this energy function as follows (see Fig. 1):

$$b_{ij} = \beta \quad (3)$$

$$W_{ij,mn} = \begin{cases} -\omega & \text{if } (d_{i,j}, d_{m,n}) \in C_a(X_i, X_m) \\ -\eta & \text{if } i = m, j \neq n \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

where  $\beta$  and  $\omega, \eta > \beta$  are positive constants. The first set of terms in Eqn. (4) implement the constraints  $C_a$ , i.e. if a pair of assignments is forbidden by any constraint, then there is an inhibitory link between the corresponding neurons. The second set of terms represents the condition that at most one value can be assigned to each variable.

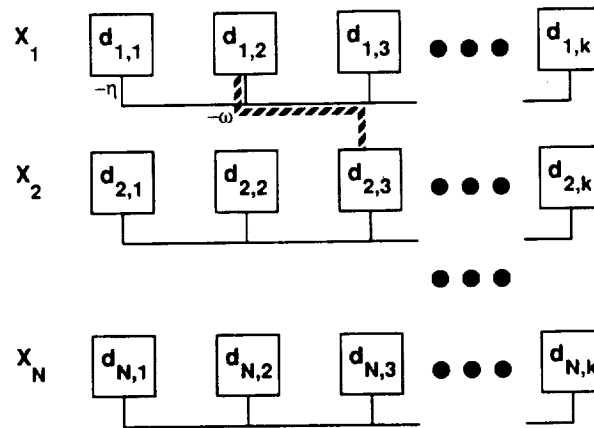


Figure 1. A Hopfield network for a binary CSP: variables are represented by rows, value assignments by neurons on each row (labelled by the domain value they represent). Here it is assumed that each variable  $X_i$  can assume one of  $k$  values. The network includes a set of symmetric inhibitory links that permit only one value to be assigned to each variable (solid lines) and another set that represents the binary constraints (one example is shown as a heavy dashed line).

If the network update algorithm Eqn. (2) is applied to this problem it is quickly found that, while the network sometimes converges to an assignment for all  $N$  variables, it frequently comes to rest in a stable state with  $n < N$  neurons active: these are local minima of the energy function. The GDS network introduces a way to escape local minima that is especially well-suited for discrete networks: the asymmetric coupling of the main network to an auxiliary network (Fig. 2). The auxiliary network, which we call a guard network, is designed to enforce an additional important condition of the problem, namely that when a solution is found, every variable must have an assigned value. When this condition is enforced, states with  $n < N$  neurons active are no longer stable, and so the network continues to evolve.

The guard network consists of an additional  $N$  neurons, one for each variable which must have an assigned value. A guard neuron with bias  $b_i^g$ , input  $x_i^g$ , and output  $y_i^g$  is connected to each neuron on the row  $i$  that it guards. The input to the guard is  $x_i^g = -\theta \sum_j y_j$ , while the contribution by the guard to the input of neuron  $ij$  is  $\phi y_i^g$ . If we choose the guard bias to be  $b_i^g = \gamma > 0$  and choose  $\theta > \gamma$  and  $\phi > 0$  sufficiently large, then the guard on row  $i$  will fire only when no neurons on row  $i$  are firing. When the guard fires, a large positive value  $\phi$  is added to the input of each neuron on the row: if  $\phi$  is chosen to be large enough to overcome the

effect of any number of inhibitory links, then any neurons on the row can transition from off to on, thereby reducing the energy of the network. Thus local minima due to the absence of any firing neurons on a row are eliminated. The price paid for this desirable feature is that the symmetry of the connection matrix for the combined network is lost, and thus convergence to a stable state is no longer guaranteed. In practice some stopping criterion must be specified, which may be problem-dependent.

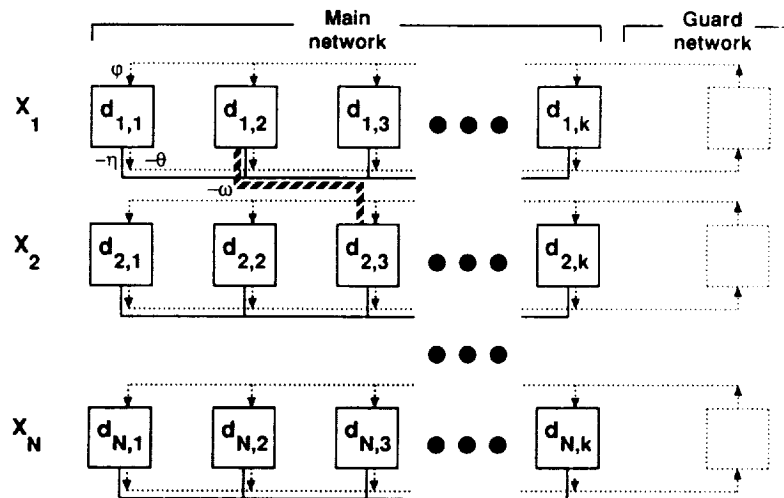


Figure 2. The GDS network: the network of Fig. 1 is coupled asymmetrically to a guard network to enforce the condition that each variable must have an assigned value (dotted lines).

We have found that it is most effective to update the guard network synchronously with transitions on the main board, i.e. each guard's output is always maintained consistent with its input according to Eqn. (2). This essentially treats the guards as a separate network which runs on a faster timescale than the main network. We have also found that random selection of which neuron in the main network to examine next is much less effective than selecting at random one *set* of neurons which are monitored by one guard neuron, then changing the state of the neuron in the set whose output is "most inconsistent" with its input (if any). That is, we select the neuron with the maximum value of either

$$x_{ij} \text{ if } y_{ij} = 0 \text{ and } x_{ij} \geq 0, \text{ or } |x_{ij}| \text{ if } y_{ij} = 1 \text{ and } x_{ij} < 0, \quad (5)$$

with ties broken arbitrarily.

The initial state of the network is an important consideration. If for a particular CSP there is some heuristic which can identify variable assignments which are "likely" to be part of a solution, then these can be used to specify the initial network state. If, as is often the case, no such assignments are known, then it is appropriate to start the network with all neurons in the off state  $y_{ij} = 0$ . In either case, the initial state will usually have nearly all neurons in the off state. This leads to the observation that the connections  $W_{ij,mn}$  need not be pre-computed and stored, but may be calculated only when neuron  $mn$  first transitions from off to on. (If the connections can be computed efficiently enough then it may not even be effective to store them at all). This can permit a large reduction in storage requirements: even though the number of possible connections may be large, only a small fraction may be instantiated during any given set of runs of the network.

An example of the GDS network's performance is provided by the well-studied  $N$ -queens problem of placing  $N$  queens on an  $N \times N$  chessboard, one on each row, so that no queen threatens another. This can be represented as a binary CSP with  $N$  variables representing the chessboard rows and  $N$  values representing the columns in



which the queens are placed. The connections  $W_{ij,mn}$  encode the constraints that no two queens can threaten each other along columns or diagonals. Row threats are automatically disallowed since variables can only have one assigned value.

$N$ -queens has been used as a model problem in several studies of improvements to regular backtracking search: see especially Stone and Stone [13] who conducted an investigation of backtracking and most-constrained search for  $N$  up to 96. They suggest that backtracking has exponential, and most-constrained search has polynomial time complexity over the range of  $N$  they studied, but they note that they were unable to find solutions in a "reasonable amount of time" for  $N = 97$ . A continuous neural network representation of the 8-queens problem was investigated in [5].

Solutions to the  $N$ -queens problem are easily found by the GDS network. In Fig. 3 is plotted the median number of neuron transitions ( $0 \rightarrow 1$  and  $1 \rightarrow 0$ ) required for the network to converge to a solution (estimated from a large number of runs) versus linear board-size  $N$ . The result is linear in  $N$  for large  $N$  as shown by the straight line fit. Note that a minimum of  $N$   $0 \rightarrow 1$  transitions is required to proceed directly from the "empty board" initial state ( $y_{ij} = 0$  for all  $ij$ ) to a solution with no "wandering". The surprising result is that only a proportionately small number of *excess* transitions beyond this minimum is required to find solutions: empirically this excess is found to be about  $0.16N$ . To check that this behavior holds for very large  $N$  we have run the network with  $N$  as large as 1024. This corresponds to a main network containing  $N^2 \approx 10^6$  neurons, with  $> 10^9$  potentially non-zero connections. A solution to the  $N = 1024$  problem was found to require only 1196 transitions and required a wall-clock time of less than 12 minutes on a 16Mb TI Explorer II workstation.

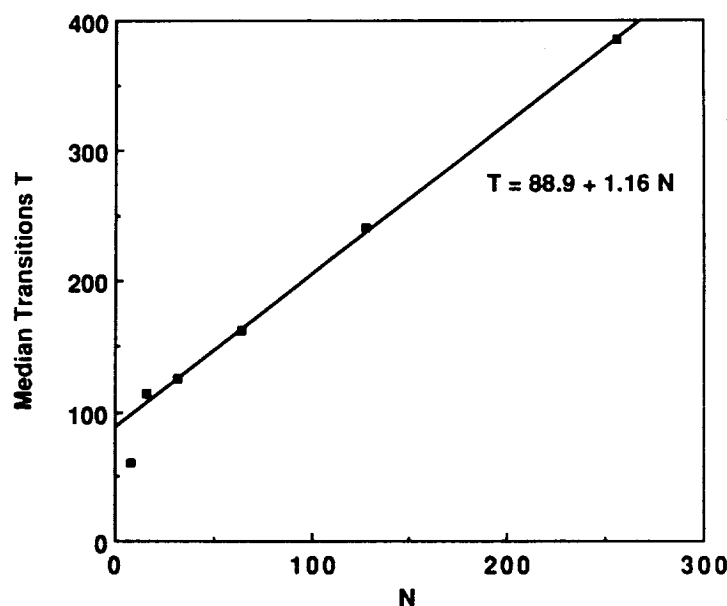


Figure 3.  $N$ -queens: Median number of transitions to convergence vs. linear board-size  $N$ .

The expected time complexity of the GDS network on the  $N$ -queens problem is  $O(N^2)$ , since the expected number of transitions to convergence is (empirically)  $O(N)$  and each transition requires adding a connection weight to the inputs of  $O(N)$  inhibited neurons (and the overhead associated with each transition is also  $O(N)$ ). The space complexity of the network is  $O(N^2)$ , even though the number of non-zero connections is  $O(N^3)$ . Further results of the GDS network on  $N$ -queens and other CSPs is provided in [10].

### 3 The GDS Learning Algorithm

It has long been known that pre-processing constraints in CSPs can lead to dramatic improvements in the effectiveness of backtracking search [1,2]. These techniques, known as consistency methods, are based on the deduction of additional constraints from those explicitly provided. These additional constraints can be exploited in backtracking search to avoid repetitively exploring sets of assignments that cannot be part of any solution. While these techniques have generally been applied before search begins, Dechter [12] has shown how they can be applied during the search process to provide a kind of "learning while searching". An analogous learning process can be defined for the GDS network by exploiting the fact that network connections need not be instantiated until they are needed, i.e. when a neuron participating in a constraint transitions from off to on. Learning can be based on instantiated connections only, leading to changes in the network biases and connection strengths that improve the performance of the network from one run to the next.

The GDS learning algorithm we have developed is independent of the problem represented by the network. It operates as a separate module which analyzes the results of one or more "training" runs to update the network bias values, connection strengths, or both. Training consists of the following series of steps which can be repeated as often as desired:

1. Starting with all neurons off ( $y_{ij} = 0$  for all  $ij$ ), run the network for a fixed number of transitions  $T_{train}$  and record the connections for each neuron  $ij$  which transitions from 0 to 1. Denote the set of all neurons which have transitioned from 0 to 1, since the network was initialized, as ON.
2. Reset all neurons to their off state.
3. For each neuron  $ij$  in ON with  $x_{ij} \geq 0$ , set its state to on ( $y_{ij} = 1$ ) and turn off all others. Update the inputs of any other neurons  $mn$  based on the recorded connections  $W_{mn,ij} \neq 0$ . If  $mn$  is in ON and  $x_{mn} \geq 0$  and  $x_{mp} < 0$  for  $p \neq n$ , then set  $y_{mn} = 1$  and update inputs again. Repeat until no further changes occur.
  - update biases: if there is any row  $m$  such that  $x_{mn} < 0$  when  $b_{mn} \geq 0$  for all  $n$ , then set the bias of  $ij$  to some value  $b_{ij} < -\phi$  (effectively removing  $ij$  from the network, i.e. deleting  $d_{ij}$  from  $D_i$ ).
  - update connections: if there is any  $mn$  such that  $x_{mn} < 0$  when  $b_{mn} \geq 0$ , then record the connection coefficient  $W_{mn,ij} = -\omega$ . This represents an induced constraint between  $X_i$  and  $X_m$  indicating that  $d_{ij}$  and  $d_{mn}$  are incompatible assignments and cannot be part of any solution.

Updating only the biases corresponds to a partial arc-consistency algorithm where only instantiated connections are considered. Updating the connection weights corresponds to partial path-consistency, i.e. the recording of additional induced constraints. These two update schemes correspond in Dechter's nomenclature to "first-order" and "second-order" learning, respectively. Note that updating the connections as described above does not correspond to full path-consistency even on the set of instantiated connections, since additional constraints could possibly be induced by those discovered during a training step. Thus the computational effort expended in training is much less than that required to perform full arc- or path-consistency [14,15].

We have compared the results of applying this learning algorithm to the results obtained from running the network on only those constraints provided explicitly in the definition of the problem. Two CSPs have been used in this investigation: the random CSP used by Dechter and Pearl in their study of Advised Backtracking [4], and the Zebra problem used by Dechter in her investigation of learning in backtracking search [12].

#### 3.1 The Dechter-Pearl Problem

This problem is one of a family of random CSPs [16] specified by four parameters: the number of variables  $N$ , the number of values  $k$  each variable can assume, the probability  $p_1$  of having a constraint between any pair of variables, and the probability  $p_2$  that a constraint allows a given pair of values. The behavior of the GDS

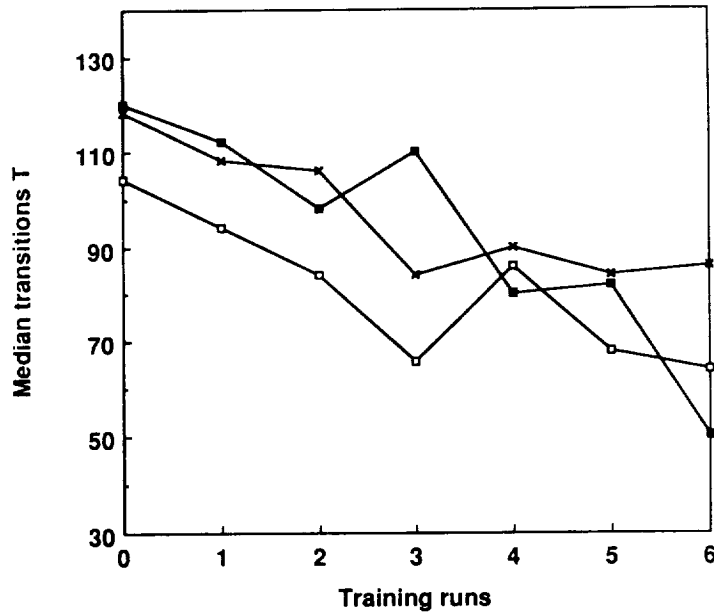


Figure 4. Dechter-Pearl CSP: median number of transitions to convergence vs. number of first-order training runs for three randomly-generated problem instances (filled squares, open squares, and crosses).

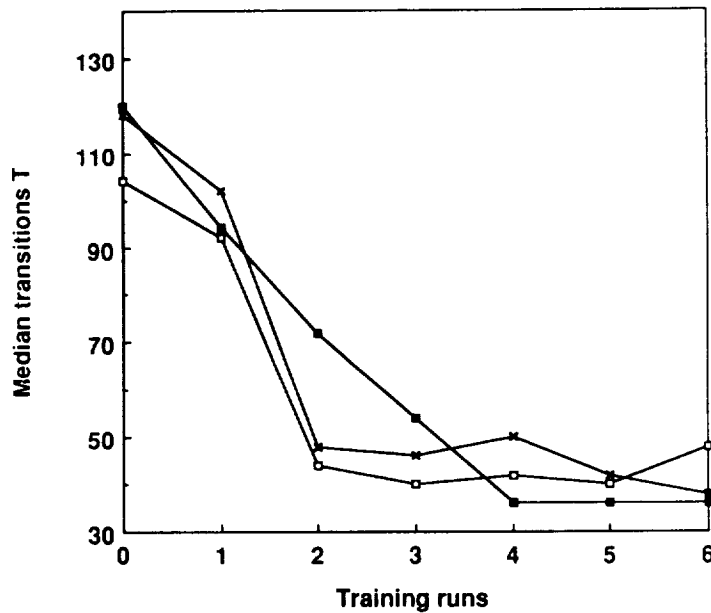


Figure 5. Dechter-Pearl CSP: median number of transitions to convergence vs. number of second-order training runs for the same problem instances as Fig. 4.

network on this problem for  $k = 5$ ,  $p_1 = 0.5$ , and  $p_2 = 0.6$  was reported in [10] for a range of  $N$  between 30 and 120. The median number of transitions  $T$  required for the network to converge was found to be linear in  $N$ :  $T \cong 35 + 2.5N$ .

Here we consider the case  $N = 30$  and investigate the effectiveness of the bias and connection learning algorithms on the performance of the network. Three randomly-generated problem instances were generated and subjected

to a variable number of training sessions ranging from one to six. One set of runs consisted of bias updates only (first-order learning); the other consisted of both bias and connection updates (second-order learning). Each training run was arbitrarily limited to  $T_{train} = N$  transitions. Each series of training runs was started from the explicit constraints only, i.e. there is no correlation of the results as the number of training runs increases.

The results of first-order learning are plotted in Fig. 4 which shows the median number of transitions required for the network to converge to a solution vs. the number of training runs. Note that a minimum of  $N = 30$  transitions is required to proceed directly from the initial network state  $y_{ij} = 0$  to a solution. It can be seen that there is an approximately steady decrease in the median number of transitions, from about 115 with no training to an average of about 65 with six training runs.

Second-order learning (Fig. 5) shows a more significant performance improvement with the first few training runs, but little further improvement with additional training. After only three training steps the median number of transitions has decreased from 115 to an average of about 45.

### 3.2 The Zebra Problem

This significantly harder problem was described by Dechter (see Appendix II of [12]) and was used in her study of learning during backtracking search. The problem consists of  $N = 25$  variables, each with 5 possible values. The GDS network without learning converges to a solution only about 10% of the time when limited (arbitrarily) to  $9N$  transitions. Although first-order learning makes only a marginal difference in the performance of the network, second-order learning shows a dramatic improvement. Fig. 6 shows the probability of convergence in

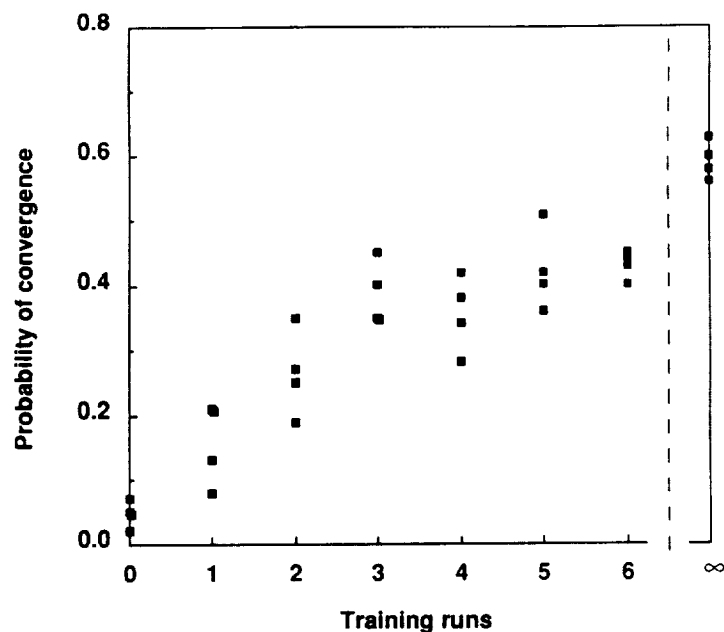


Figure 6. The Zebra Problem: probability of convergence in  $9N$  transitions vs. number of training runs, with " $\infty$ " representing a fully path-consistent version of the problem.

$9N$  transitions vs. number of second-order training runs. The results for " $\infty$ " are for a fully path-consistent version of the problem and represents the best that can be achieved by increasing the amount of training. Even a small number of training runs can clearly improve the network performance by a significant margin.

## 4 Discussion

The behavior of the GDS network can be likened to a stochastic backtracking algorithm which implements a number of "heuristics" to expedite search. Stochastic, in contrast to regular backtracking, means that the order of instantiation of variables is not pre-determined: backtracking makes a systematic exploration of the search tree, while the network stochastically probes the tree in directions that tend to minimize the network energy. Since the network permits temporary inconsistencies in variable assignments at any point until it converges, it can make "lateral jumps" in the search tree to escape from sets of assignments that cannot be consistently extended. These jumps appear to be useful in discovering consistent assignments, although their effectiveness depends on the detailed structure of the search tree (as evidenced by the results on 3-Colorability in [10]).

The heuristics intrinsic to the network come into play when a partial instantiation cannot be consistently extended. This corresponds to encountering a deadend during backtracking search. These network heuristics cannot be strictly isolated (since extending partial assignments and backing out of deadends are simultaneous competing processes), but they can be loosely compared to those developed to improve the behavior of backtracking algorithms:

- backjumping: when the network encounters a deadend it will randomly select an uninstantiated variable and assign it a value which is certain to be inconsistent with one or more previously made assignments. This entire set of inconsistent assignments is at once subject to revision: at least one will eventually be retracted. This corresponds closely to the backjumping or "go back to cause of failure" heuristic which is known to improve the performance of regular backtracking, but is somewhat more general in that any variable with no permitted assignments can be considered the "failure", and any variable with which it is inconsistent can be considered the "cause".
- value selection: when the network extends a partial instantiation by assigning a value to an unassigned variable, any value not forbidden by some constraint is equally likely to be chosen. However, at a deadend, values are selected which are least inhibited by any current assignments (since the neuron input is proportional to the number of constraints that forbid the assignment). This represents a kind of value selection heuristic which undoes a *minimal* set of previous assignments in order to escape from the deadend. Only value assignments that participate in such minimal sets will be made by the network update algorithm Eqn. (5).

The GDS network is a general constraint satisfaction search method, encoding no domain knowledge other than value and value-pair inhibitions. Nevertheless the convergence of the network on some classes of problems is remarkably fast. This, along with the ease with which the network can be set up for new problems, makes it an attractive approach for some classes of large CSPs. We have shown here that the performance of the network can be significantly improved by adding a "learning" module that analyzes the results of one or more training runs and updates the initial values of the neuron biases and connection strengths. The learning algorithm is independent of the problem represented by the network. In terms of the network heuristics discussed above, the effectiveness of this type of learning is due to the resulting increase in the sizes of minimal sets of inconsistent variables. As a result, inconsistent sets are encountered after fewer transitions, and longer and more relevant "jumps" in the search space are made possible. For some types of problems this dramatically improves the speed of convergence of the network.

## References

- [1] Montanari, U.: 1974, "Networks of Constraints: Fundamental Properties and Applications to Picture Processing", *Information Sciences* 7, 95-132
- [2] Mackworth, A.K.: 1977, "Consistency in Networks of Relations", *Artif. Intell.* 8, 99-118
- [3] Freuder, E.C.: 1982, "A Sufficient Condition of Backtrack-free Search", *J. ACM* 29, 24-32

- [4] Dechter, R., Pearl, J.: 1988, "Network-based Heuristics for Constraint-Satisfaction Problems", *Artif. Intell.* **34**, 1-38
- [5] Tagliarini, G.A., Page, E.W.: 1987 "Solving Constraint Satisfaction Problems with Neural Networks", *Proc. IEEE First Internat. Conf. on Neural Networks, San Diego* **3**, 741-747
- [6] Dahl, E.D.: 1987, "Neural Network Algorithm for an NP-Complete Problem: Map and Graph Coloring", *Proc. IEEE First Internat. Conf. on Neural Networks, San Diego* **3**, 113-120
- [7] Hopfield, J.J.: 1982, "Neural networks and physical systems with emergent collective computational abilities", *Proc. Nat. Acad. Sci.* **79**, 2554-2558
- [8] Kirkpatrick, S., Gelatt, C., Vecchi, M.: 1983, "Optimization by simulated annealing", *Science* **22**, 671-680
- [9] Ackley, D.H., Hinton, G.E., Sejnowski, T.J.: 1985, "A Learning Algorithm for Boltzmann Machines", *Cognitive Science* **9**, 147-169
- [10] Adorf, H.-M., and Johnston, M.D.: 1988, "A Discrete Stochastic "Neural Network" Algorithm for Constraint Satisfaction Problems", submitted.
- [11] Hinton, G.E.: 1987, "Connectionist Learning Procedures", CMU Tech. Report CMU-CS-87-115 (Ver. 2), Department of Computer Science, December 1987.
- [12] Dechter, R.: 1986, "Learning While Searching in Constraint Satisfaction Problems", *Proc. AAAI-86, Fifth Nat. Conf. Artif. Intell., Philadelphia* **1**, 178-183
- [13] Stone, H.A., Stone, J.M.: 1987, "Efficient search techniques — An empirical study of the *N*-Queens Problem", *IBM J. Res. Devel.* **31**, 464-474
- [14] Mackworth, A.K., Freuder E.C.: 1985, "The Complexity of Some Polynomial Network Consistency Algorithms for Constraint Satisfaction Problems", *Artif. Intell.* **25**, 65-74
- [15] Mohr, R., Henderson, T.C.: 1986, "Arc and Path Consistency Revisited", *Artif. Intell.* **28**, 225-233
- [16] Haralick, R.M., Elliott, G.L.: 1980, "Increasing Tree Search Efficiency for Constraint Satisfaction Problems", *Artif. Intell.* **14**, 263-313

# INTEGRATING PLANNING, EXECUTION, AND LEARNING

Daniel R. Kuokka

Computer Science Department<sup>1</sup>  
Carnegie Mellon University  
Pittsburgh, PA 15213

## Abstract

To achieve the goal of building an autonomous agent, the usually disjoint capabilities of planning, execution, and learning must be used together. This paper describes an architecture, called MAX, within which cognitive capabilities can be purposefully and intelligently integrated. The architecture supports the codification of capabilities as explicit knowledge that can be reasoned about. In addition, specific problem solving, learning, and integration knowledge is developed.

## 1. Introduction

The expense, isolation, danger, and uncertainty involved in space research vividly points out the need for robust autonomous robots. However, the current generation of intelligent systems only present a subset of the requisite behavior. For an intelligent system to be a competent autonomous agent, many different cognitive capabilities must be solidly integrated. For example, an autonomous system is more than a planner that generates an answer given a precise problem specification; it must actually take actions dictated by its reasoning. However, it is not sufficient merely to give a plan to a simple execution system; the unpredictability of the world makes it very unlikely that the plan is correct or complete. A system must be able to suspend planning or execution in order to seek needed information. However, the acquisition of knowledge, itself, may require planning and execution. Thus, there is a complete interdependence between the various cognitive capabilities.

These issues stem from the relaxation of two assumptions commonly made in problem solving: the availability of complete knowledge, and a static environment. Complete knowledge is fundamental to the plan-then-execute paradigm assumed by many problem solvers. However, it is an assumption that is simply invalid in many cases. The ability to take action based on incomplete information, and to intentionally acquire new knowledge, is fundamental if a system is to operate robustly in real environments. Another complication is that real-world environments tend to be dynamic. A successful agent must always be aware of the external world, and it must be able to suspend any task in favor of a more urgent task. The problem of integrating planning and execution in a dynamic world is even more complex if consideration is given to the fact that reasoning, itself, is not a "free" activity. Reasoning requires time, and it is not always safe to assume that a system has as much time to think as required. A successful autonomous agent must

---

<sup>1</sup>This research was supported in part by ONR grants N00014-79-C-0661 and N0014-82-C-50767, DARPA contract number F33615-84-K-1520, NASA contract number NCC 2-463, and a grant from the Hughes Aircraft Corporation. The views and conclusions contained in this document are those of the author alone and should not be interpreted as representing the official policies, either expressed or implied, of the US government, ONR, DARPA, NASA, or the Hughes Aircraft Corporation.

be able to rationally control its reasoning, just as it must rationally control its physical capabilities.

This paper describes work in progress aimed at *integrating* previously separate capabilities into a unified autonomous system that can function in an unknown and dynamic environment. The focus is the synergistic interaction of the components, rather than better individual components. There are two main components of the research. First is the development of an architecture, called MAX, that permits the principled, knowledge-based integration of complex capabilities via meta-level reasoning. This is discussed in section 3. Second is the codification of knowledge that intelligently interleaves each capability. This is presented in section 4. Section 5 describes an extended example in which all parts of the system work together. However, before delving into MAX, a brief survey of other work is presented.

## 2. Related Work

Prior research has tended to focus on various subproblems of building autonomous agents, such as planning, learning, and vision, under the principle of divide and conquer. There has been little work on systems that integrate such behavior. However, there has recently been an increase of interest in the high-level control of autonomous agents, as well as cognitive architectures that cover, in principle, a broad range of capabilities. These systems can be roughly organized according to their reliance on explicit reasoning.

At one end of the spectrum are those systems, such as Pengi [1], that propose situated activity as the mechanism behind higher level planning. The research of Brooks [3] emphasizes the use of a hierarchy of behavioral components, each of which uses perceptual information directly to produce some behavior. Kaelbling [8] also tends toward such an architecture, but she also suggests the need for explicit planning. Another step toward explicit reasoning is represented by the Procedural Reasoning System system of Georgeff et al [6] which does pattern-directed invocation of canned procedures. These systems posit, to varying degrees, that high level planning is not required, as it can emerge from lower-level behavior. This thesis has not yet been adequately proven.

Moving away from robot control architectures, a number of systems propose a relatively weak integration of intelligent modules. This model is exemplified by plan-then-execute systems such as NOAH [13] and SIPE [16]. Another form of loose integration is the black-board architecture, used by systems such as BB1 [7] and Codger [15]. In general, the top-level control structures used by these loosely integrated systems only address a subset of the integrated behavior required of autonomous agents.

Strongly integrated systems are represented by the set of general architectures intended to be applicable to any task. Cognitive architectures such as ACT\* [2] and SOAR [9] have achieved a fair degree of success in this regard. However, they have not been applied to the task of rationally integrating various intelligent capabilities in a dynamic environment. Furthermore, the assumptions imposed by the added constraint of cognitive plausibility may make such integration difficult. Another candidate architecture is PRODIGY [11], which provides a variety of base-level reasoning and learning mechanisms. In fact, the work described in this paper is an effort to evolve PRODIGY into a competent meta-level reasoner.

Further along the spectrum are systems designed explicitly for meta-level reasoning. Examples include MOLGEN [14], MRS [5], and Theo [12]. These systems are capable of very elaborate reasoning, but their application to a real-time, irreversible environment is problematical.

Since there hasn't been much work on meta-level architectures intended for autonomous agent control,



there is very little in the way of meta-level knowledge required by such a system. Only pieces have been investigated. The work on execution monitoring includes some techniques for coping with unexpected situations, but only in restricted environments. Carbonell and Gil [4] have developed techniques to guide experimentation to learn operators, and Eurisko [10] also performs some experimentation. However, the higher level issues of intelligently integrating experimentation into an autonomous agent remain unanswered.

### 3. The MAX Architecture

The main goal of the MAX architecture is to allow cognitive capabilities (e.g., planning and experimentation) to be flexibly controlled. A powerful way to accomplish this is to enable the system to reason about mental actions (perform meta-level reasoning) as well as reason about physical actions (perform base-level reasoning). This, in turn, requires that system's cognitive capabilities be encoded as explicit knowledge. It is this approach that underlies the design of MAX.

To elaborate, consider a traditional system in which the capabilities are hardwired into the kernel; the capabilities can be applied only according to the foresight of the designer and the flexibility of the algorithm. Undoubtedly, unforeseen situations will crop up with which the system cannot cope. Now consider a system in which its capabilities are explicit knowledge. Such a system can use its entire reasoning power to flexibly apply its capabilities. The system is no longer restricted to some hardwired interaction between planning, execution, and learning. Furthermore, since the capabilities are encoded just like any base-level knowledge, their implementation can use arbitrary amounts of knowledge to obtain expert performance, and the capabilities can be modified by the system.

Encoding cognitive capabilities as knowledge requires that the knowledge representation be sufficiently powerful, and that the system have the ability to execute explicit knowledge. It is these two requirements that the MAX architecture is designed to meet. Section 3.1 describes MAX's knowledge representation, which is powerful enough to implement cognitive capabilities while remaining simple enough to reason about. Next, section 3.2 describes the control structure, which allows explicit knowledge to be executed.

#### 3.1. Knowledge Representation

The foundation of MAX's knowledge representation is first order predicate logic augmented with a single data structure, called an *lframe* (for logic-frame). An *lframe* can best be described as an independent, explicit logical database, or state. Therefore, in contrast to traditional logic, where terms are atomic symbols only, terms in MAX can also be structures representing entire states. Since a state is a collection of logical assertions, an *lframe* is a recursive data structure, not unlike frames. However, there are two features of *lframes* that produce the real leverage of the representation: the capability to represent with one *lframe* an entire sets of states, and the presence of a set of high-level relations over *lframes*.

Consider figure 3-1 which shows a single *lframe* representing the definition of a Strips-style operator. Brackets denote *lframes*, and parentheses denote logical assertions, or literals. Within the operator, there are a number of assertions that specify its components: namely the input parameters, preconditions, adds, and deletes. Furthermore, each component is, itself, an *lframe* that represents a state (or complement of a state) of the blocks world. Note the **\$vars** assertion, which declares a set of variables within the operator. The variables allow the *lframe* to represent an infinite number of operators, corresponding to the various instantiations of the variables. Also notice that the variables declared within the operator as a whole are referenced within the subcomponents of the operator. This constrains the values of subcomponents of the

operator that can co-occur, insuring the operator remains valid (i.e., if the precondition matches a particular object, the operator should not delete a different object from the state). The use of lexical variables is an important feature of lframes, allowing the succinct representation of an entire set of states.

```

[($vars ?block1 ?block2)
 (parameters [(topblock ?block1) (bottomblock ?block2)])
 (precondition [(holding ?block1)
                (clear ?block2)])
 (delete [(holding ?block1)
          (clear ?block2)])
 (add [(on ?block1 ?block2)])]

```

Figure 3-1: Definition of the put operator

The other main feature of lframes is the definition of a set of high-level relations, most notably **match**, **intersection**, **union**, and **difference**. These relations are basic to the reasoning process, just as **on** is basic to the blocks world. These basic operations allow reasoning to be represented naturally as explicit knowledge. Consider figure 3-2, which gives the definition of operator **apply**. This operator states the effects of applying a *subject* operator, for example, **put**. The preconditions decompose the subject operator into its components, verify that the preconditions match the current state, and specify the relationship between the old state, deletes, adds, and new state. The delete and add of **apply** then specify the changes to the state of the planner. The structure of the operator is identical to the base-level operator, only the relations have changed.

```

[($vars ?op ?del ?add ?state1 ?state0 ?state2 ?oldplan ?newplan)
 (parameters [(operator ?op)])
 (precondition [(state ?state1)
                (match ?op [($vars ?pre)
                             (precondition ?pre)
                             (match ?state1 ?pre)
                             (delete ?del)
                             (add ?add)])
                (difference ?state1 ?del ?state0)
                (union ?state0 ?add ?state2)
                (plan ?oldplan)
                (postpend ?oldplan ?op ?newplan)])
 (delete [(state ?state1)
          (plan ?oldplan)])
 (add [(state ?state2)
       (plan ?newplan)])]

```

Figure 3-2: Definition of the apply operator

In summary, lframes are vital to the MAX architecture for several reasons. Lframes are an appropriate data structure for representing the basic fodder of reasoning, namely states of knowledge. The capability to introduce variables within lframes allows complex sets of states to be succinctly represented. Such a capability is not present in either traditional logic or frame-based representations. The definition of basic relations over lframes allows the consequences of knowledge to be represented without resorting to representational details (i.e., traditional logic could be used to represent the apply operator, but the result would be the implementation of a theorem prover in logic.) These features make it practical to *represent* complex reasoning, a necessity in the rational integration of high-level capabilities.

### 3.2. Control Structure

In addition to the knowledge representation, the other main component of the MAX architecture is the control structure. The main goal is to allow the control of the agent to be based on explicit knowledge. This supports a very flexible flow of control, avoiding the problem of control being hardwired into the kernel. Indeed, there must still be some kernel that interprets the knowledge, but once the powerful capabilities are removed, the kernel can be very simple. In other words, the kernel of MAX is just powerful enough to *execute* knowledge rather than reason about knowledge.

The basic unit of the control structure is called a *task*, which is an lframe structure. A task can be thought of as an explicit representation of a production system with two exceptions. First, the "productions" are divided into operators and control rules representing what can be done and when to do it, respectively. This allows a variety of control to be applied to a fixed set of actions (i.e., even though actions are determined by the physical capabilities of the agent, when to do them is flexible). Second, an operator (called a *complex* operator) can invoke a subtask that does an arbitrary amount of processing, similar to operator implementation problem spaces in SOAR. This allows an agent to reason about abstract actions, such as building a bridge or creating a plan that achieves a goal.

Consider the example of figure 3-3, which shows a planning task. The operators and control rules correspond to traditional productions. The state corresponds to working memory. In this example, the planning state consists of a blocks world goal, a current blocks world state, the plan being formed, and a domain theory specifying the legal operators for the blocks world. This illustrates how the same language can be used both to implement reasoning, and to represent knowledge (the object of reasoning). Finally, the **pending** assertion indicates that this task was invoked by executing a complex operator within a higher-level task, called **solve**. When the planning task is finished, the results are added to the solve task, which is then resumed.

```
[(rule apply-satisfied-operator [(condition [...])
                                (action subgoal [...])])
 (rule subgoal-on-goal-state-diff [...])
 ...
 (operator apply [(precondition [...])
                  (delete [...])
                  (add [...])])
 (operator subgoal [...])
 ...
 (state [(goal [(on block1 block2)])
         (state [(on-table block1)
                  (on-table block2)])
         (plan [(elt 1 pick [...]) (elt 2 put [...])])
         (domain [(operator put [...])
                  (operator pick [...])])])
 (pending solve [(rule plan-when-difficult-goal [...])
                 ...
                 (operator plan [...])
                 ...
                 (state [...])])])
```

Figure 3-3: Task implementing a planner

The choice to build the MAX control structure on a production system was inspired by the successes encountered in the world of expert systems. Also, productions are a natural means of implementing a highly reactive system, a requirement for any autonomous agent. However, MAX is unique in that it

unifies production and working memory, in addition to process memory (tasks actually represent the execution state). This gives MAX extreme flexibility to reason about and modify its own behavior, which means nearly all control decisions can be considered explicitly. Finally, the separation of various capabilities into different tasks results in a modular system, a large advantage from the practical standpoint of system development.

#### **4. Domain Knowledge**

The MAX architecture provides a framework in which capabilities can be encoded and controlled. This section outlines the kind of capabilities that can be encoded. Each capability of the agent, from the most primitive (such as simple motion) to the very complex (such as planning), is encoded as a domain. A domain is defined as a set of operators and a set of control rules. Therefore, a task is a snapshot of the execution of a domain. Furthermore, since complex operators result in the execution of a subtask, there is a domain associated with each complex operator. Conversely, a complex operator represents the external specification of a domain.

Table 4-1 gives an overview of the key domains required to implement and integrate intelligent capabilities. Each row specifies the name of the domain (which corresponds to the complex operator that is used to invoke it), the key operators, selected control rules, and the main objects about which the domain reasons. Notice that many domains use complex operators corresponding to other domains; thus, the capabilities of the system are applied recursively.

The function of the select domain is to choose between various potential goals or activities. The select domain consists of two operators, solve a goal (described below) and run a heuristic procedure (which simply executes a specified domain). In addition, there are rules that specify when a particular operator should be applied. In general, goals concerning robot safety should be considered first, followed by primary goals, followed by background goals. Also, procedures should be used when available. However, there are likely to be interactions requiring specific exceptions. This illustrates the heuristic nature of the knowledge required in such domains; quite often there is no simple algorithm that provides the requisite behavior.

The solve domain provides a rather unique but powerful capability, namely that of intelligently selecting the problem solving approach used for a given goal. For example, the solve domain can choose between forming and executing a plan, specializing and executing a canned plan, and executing a heuristic procedure. The choice of problem solving tactic can be based on the presence of the requisite knowledge plus heuristics about the particular domain and problem. Finally, the solve domain includes the option of intentionally learning more about the environment or a domain theory. The importance of this is discussed below.

Within the plan domain, there are two main operators corresponding to the two main actions of means-ends analysis: subgoal on an operator and applying an operator. Each of these operators affects the state of the planner by modifying the hypothetical state, the goal, or the partial plan. Notice that the apply operator is exactly that presented in figure 3-2. Unlike many problem solvers, a particular planning algorithm is not built in to the system. Alternate domains could be added that implement forward chaining or various forms of non-linear planning. Knowledge within a higher level domain would then be required to select the most appropriate technique.

The execute domain is, on the surface, rather simple. It simply takes a plan and attempts to execute it.

DOMAIN	OPERATORS	RULES	STATE
<b>select</b>	<b>run</b> <b>solve</b>	<b>run-emergency-procedure</b> <b>solve-given-goal</b>	<b>current-state</b> <b>potential-goals</b>
<b>solve</b>	<b>plan</b> <b>execute</b> <b>run</b> <b>explore</b> <b>experiment</b>	<b>plan-when-difficult-goal</b> <b>execute-when-plan-exists</b> <b>run-heuristic-procedure</b> <b>explore-when-info-needed</b> <b>experiment-when-faulty-ops</b>	<b>goal</b> <b>current-state</b> <b>base-domains</b>
<b>plan</b>	<b>apply</b> <b>subgoal</b>	<b>apply-satisfied-operator</b> <b>subgoal-on-goal-state-diff</b> <b>exit-when-unsatisfied-axiom</b>	<b>goal</b> <b>current-state</b> <b>base-domain</b> <b>plan-so-far</b>
<b>execute</b>	<b>step-plan</b> <b>solve</b>	<b>step-plan-when-as-expected</b> <b>subgoal-when-discrepancy</b> <b>exit-when-large-discrepancy</b>	<b>plan</b> <b>expected-state</b> <b>perceived-state</b>
<b>explore</b>	<b>search-loc</b>	<b>search-inferred-location</b>	<b>needed-info</b> <b>current-model</b> <b>locs-searched</b>
<b>experiment</b>	<b>apply-op</b> <b>modify-op</b>	<b>apply-op-in-different-state</b> <b>modify-op-when-model-correct</b>	<b>expected-state</b> <b>perceived-state</b> <b>domain-theory</b>
<b>robot</b>	<b>move</b> <b>put</b> <b>pick</b>	<b>pick-up-before-moving</b>	<b>robot-loc</b> <b>object-loc</b>

**Table 4-1: An overview of cognitive domains**

However, this entails more than blindly stepping through the plan; the expectations at each step must be examined and compared to the actual situation. If a significant discrepancy appears, the execute task must decide how to correct the problem. For instance, if the discrepancy is slight, it is probably best to patch the plan and continue. However, if the problem is significant, it is best to exit the execute task and allow the higher level problem solving task to address the problem. This is an important option because the partial execution of the plan may have resulted in the acquisition of knowledge that modifies the assumptions underlying the plan. The rules governing the decision to patch or abort a plan are another example of the heuristic knowledge required within many capabilities.

Even though a variety of failures can occur within the execute and plan domains, the basic cause is almost always incomplete or incorrect knowledge. For example, the execution of a plan can fail if the results of the operators are not as expected, and the planning domain can fail if key pieces of knowledge about the state are missing. A robust autonomous agent must be able to cope with such situations, motivating the need for an intentional knowledge acquisition capability. Two of the domains in table 4-1 supply such a capability. First, the experimentation domain implements the technique of Carbonell and Gil [4] in which the preconditions and effects of operators are learned by performing experiments. Second, the exploration domain allows the use of a partial knowledge structure in the process of augmenting that knowledge structure. For example, knowledge of a door to another room suggests that an agent should move through the door to discover the contents.

The ability to perform intentional learning requires more than the presence of learning techniques; the invocation of learning activities must be intelligently controlled. For example, an agent with an urgent task to accomplish should attempt all known approaches before embarking on a regime of experimentation to learn the best approach. Thus, the solve domain contains heuristics that control the application of the explore and experiment operators based on the current problem solving priorities. It is this level of control that is vital to the rational integration of learning and problem solving.

Finally, in addition to implementing the cognitive capabilities of the agent, domains are also used in their traditional role of representing the base-level capabilities of the agent. There is only one base-level domain listed in table 4-1, the robot domain. It is worthy of note that complex operators can be used within base-level domains, just as in cognitive domains, to introduce useful abstractions. For example, the *move* operator could invoke a subtask in which the details of path planning and obstacle avoidance are considered. This allows higher-level planners to reason about moving as if it were a simple atomic action.

## 5. A Day in the Life (of an Autonomous Agent)

The power of the MAX architecture derives from the intelligent integration of many relatively simple capabilities. To illustrate how the aggregate behavior comes about, this section presents an extended example that exercises many aspects of the system. The example is set within a simple household robot domain. For the purpose of this example, the sensory and effectory capabilities of the robot will be rather powerful. The robot has three main effectory capabilities, move, pickup, and putdown. These capabilities correspond to the operators within the robot domain outlined in the previous section. The robot has one main sensor that returns a map of the current room. Objects within the room are never obscured, but objects in other rooms cannot be seen. This provides limitations sufficient to introduce issues due to incomplete knowledge.

The top-level goal of the robot is to fetch a wrench. However, the tool is in another room, and the robot does not know its location. Furthermore, the robot has a number of standing goals such as preserving its own safety and coping with emergencies. The initial knowledge of the robot is limited to the condition of the room it currently occupies.

The top-level domain of the robot is the select domain. Within this domain, control rules select the solve operator on the goal of fetching the tool. This choice is made because there are currently no other unsatisfied goals. The robot enters the problem solving domain wherein the lack of any applicable plan or procedure causes the plan operator to be fired. This creates a subtask that actually does the planning by applying the subgoal and apply operators. Unfortunately, since the robot does not know where the wrench is, the planning task fails, and control returns to the solve task.

Within the solve domain, the failed plan operator asserts the reason for failure, namely that the location of the wrench is unknown. This causes a rule to fire that attempts to discover the needed knowledge via the explore operator. The explore operator sets up a new task in which the goal is to discover the location of the wrench. The exploration task uses the knowledge that doors lead to other rooms to drive its explorations. As the robot searches, it adds knowledge to its state model and it maintains a separate structure indicating where it has been. The mental state of the robot at this point in time, as represented by the current task, is shown in figure 5-1.

To introduce another dimension, assume that as the robot is searching for the wrench, it discovers a fire. Fires are known to be emergencies; therefore, a rule fires that suspends the explore task, allowing a higher-

level domain to address the problem. Similarly, the solve task suspends itself since it cannot cope with an emergency; it is only concerned with solving a particular goal. The select task is returned to, where emergency heuristics choose to deal with the fire instead of fetch the wrench. This causes the invocation of a different subtask containing the fire-fighting expertise.

```

[(rule search-inferred-location [...])
 ...
(operator search-location [...])
 ...
(state [(current-model [(at desk room1)
                        (at chair room1)
                        (at hutch room2)])
        (locations-searched [(elt room1) (elt room2)])])
(pending solve
 [(rule plan-when-difficult-goal [...])
  ...
 (operator plan [...])
  ...
 (state [(goal [(at wrench room1)])
          (planning-failure [(at wrench ?anywhere)])
          ...])
        (pending select [...])])])

```

Figure 5-1: Mental state while exploring

Once the fire is extinguished, the conditions that originally caused the solve task to be applied to fetching the wrench are still present. Therefore, the robot resumes this activity, which reinvokes the explore task. Eventually, the desired knowledge is discovered, noticed by matching the exploration goal against the state, and the exploration task returns its augmented world knowledge. At this point, the robot again attempts to form a plan to achieve its goal. However, the plan is computed from the current situation, with the robot at the room in which it discovered the wrench. With the knowledge of the wrench, the plan is built successfully and asserted within the solve task.

Finally, the presence of the plan that achieves the desired goal causes a new task to be created, that of executing the plan. The execute task checks each step of the plan against the actual world state. No discrepancies are found so the plan is executed successfully. This causes the execution and solve tasks to be completed, and the select task is popped to consider the next goal.

To summarize, the main point of this example is the rational interleaving of planning, execution, and learning. The robot first attempts to plan. After planning fails, the robot performs some directed physical actions to obtain more knowledge. While pursuing this task, the robot is interrupted by a more pressing task. Once the emergency is dealt with, the robot returns to its explorations. With the additional knowledge found by exploring, the robot successfully builds a plan. Once the plan is completed, it is executed.

## 6. Summary

This paper has presented a general architecture that supports explicit reasoning about cognitive capabilities. The power of the architecture is derived from a knowledge representation that provides a data structure appropriate to reasoning, and a multi-level control structure that yields reactive knowledge-based behavior. Both components emphasize modularity and the use of abstractions as a practical requirement for implementing large systems.

In addition, a collection of knowledge was described that implements the integration of planning, execution, and learning. Among the bodies of knowledge are the problem solving domain that intelligently applies other capabilities, and several intentional learning paradigms that allow the system to actively seek knowledge based on need.

A working implementation of the MAX architecture has been completed, and the key elements of the knowledge used in the example are nearly encoded. In addition, a simple simulator has been written to update the environment and provide sensory data. The remaining work will focus on completing and augmenting the capability domains to allow more complex scenarios to be handled.

## References

- [1] Philip E. Agre and David Chapman. Pengi: An implementation of a theory of activity. In *Proceedings of the National Conference on Artificial Intelligence*. 1987.
- [2] J. R. Anderson. *The Architecture of Cognition*. Harvard University Press, 1983.
- [3] Rodney Brooks. *A Robust Layered Control System for a Mobile Robot*. Technical Report 864, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1985.
- [4] J.G. Carbonell and Yolanda Gil. Learning by experimentation. In *Proceedings of the Forth International Workshop on Machine Learning*. 1987.
- [5] Michael R. Genesereth. An Overview of Meta-Level Architecture. In *Proceedings of the Third Annual National Conference on Artificial Intelligence*, pages 119-124. 1983.
- [6] Michael P. Georgeff, Amy L. Lansky, and Marcel J. Schoppers. *Reasoning and Planning in Dynamic Domains: An Experiment With a Mobile Robot*. Technical Report 380, Artificial Intelligence Center, SRI International, 1987.
- [7] Barbara Hayes-Roth. A blackboard architecture for control. *Artificial Intelligence* 26, 1985.
- [8] L. Kaelbling. An architecture for intelligent reactive systems. In M. Georgeff and A. Lansky (editors), *Reasoning about Actions and Plans: Proceedings of the 1986 Workshop*. 1987.
- [9] J. E. Laird, A. Newell, and P. S. Rosenbloom. Soar: An Architecture for General Intelligence. *Artificial Intelligence* 33(1), 1987.
- [10] Douglas B. Lenat. EURISKO: A program that learns new heuristics and domain concepts. The nature of heuristics III: Program design and results. *Artificial Intelligence* 21(1 & 2), 1983.
- [11] S. Minton, J. Carbonell, C. Knoblock, D. Kuokka, O. Etzioni, Y. Gil. Explanation-based learning: A problem-solving perspective. to appear in *Artificial Intelligence*, 1989.
- [12] Tom M. Mitchell, John Allen, Prasad Chalasani, John Cheng, Oren Etzioni, Marc Ringuette, Jeff Schlimmer. Theo: A framework for self-improving systems. *Architectures for Intelligence*. Lawrence Erlbaum, 1989, in press.
- [13] E.D. Sacerdoti. *A Structure for Plans and Behavior*. Elsevier, 1977.
- [14] M. Stefik. Planning and Meta-Planning. *Readings in Artificial Intelligence*. Tioga, 1981.
- [15] Anthony Stentz and Yoshimasa Goto. The CMU navigational architecture. In *Proceedings of the DARPA Image Understanding Workshop*. 1987.
- [16] David E. Wilkins. *High-Level Planning in a Mobile Robot Domain*. Technical Report 388, Artificial Intelligence Center, SRI International, 1986.